Mathematical model for analyzing the structure of electrospray beams

Marc Galobardes Esteban

Grau en Tecnologies Aeroespacials (GrETA)

Directed by

Dr. Manuel Gamero Castaño
Dr. Enrique Ortega

Spring semester 2020
30th of June 2020

Declaration

I hereby certify that the material, which I now submit for assessment on the programmes of study leading to the award of Bachelor of Science Degree in Aerospace Engineering, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution.

_____

Marc Galobardes Esteban
30th of June 2020

Acknowledgements

Foremost, I would like to express my gratitude to Prof. Manuel Gamero Castaño for all the guidance, knowledge and help I received from him during these months. I really appreciate the opportunity of doing research in his lab.

I also want to thank my labmates, Albert Cisquella and Marco Magnani, for everything I learnt from them and their research experience.

I gratefully acknowledge the assistance of Prof. Enrique Ortega, my director from UPC.

This thesis would not have been possible without the support of the Balsells Foundation. My sincere appreciation to the Balsells Foundation, especially to Prof. Roger Rangel.

Last but not least, my thanks must also go to my family and friends for their support.

List of Figures

List of Tables

Contents

1 Introduction

1.1 Aim

The purpose of the model is to study the behavior of a beam of charged droplets emitted by a colloid thruster. Colloid thrusters are electrostatic accelerators of charged droplets and ions generated by electrosprays [5]. Figure 1 shows the emission of a beam of charged droplets and ions. The low thrust and high thrust stability associated with a single emitter are ideal for precision spacecraft positioning applications, as well as for primary propulsion for smallsats. This technology has been demonstrated by the Disturbance Reduction System-Space Technology 7 mission, a precursor of the Laser Interferometer Space Antenna mission (LISA) [4, 10]. However, the structure of the electrospray beam remains relatively unknown.

A good understanding of how the beam spreads is necessary to assess its interaction with spacecraft surfaces, and to design the extractor and accelerating electrodes of the thruster so that the beam impingement is minimized. Impingement is the main cause of thruster failure and being able to operate for many thousands of hours is a key requirement for space missions requiring these thrusters (e.g. LISA).

The goal of the model is to obtain a full beam simulation by coupling two different approaches: a Lagrangian model for the initial region of the beam following the emission point of the droplets, in which the expansion of electrospray beams is dominated by Coulombic repulsion of the droplets, the trajectories of droplets are integrated individually; and an Eulerian model for the rest of the beam based on the calculation of the envelopes of groups of droplets in which the overall droplet population is divided. The initial conditions required for the Lagrangian model are found experimentally whereas the initial conditions of the Eulerian model are the output of the first. The Lagrangian approach is time-dependent whereas the Eulerian approach is time-independent.

Only the model for the inner region will be discussed in this thesis. The inner

region aims to capture the correct physics of the Coulomb repulsion during the earliest stages of the spreading of the beam.

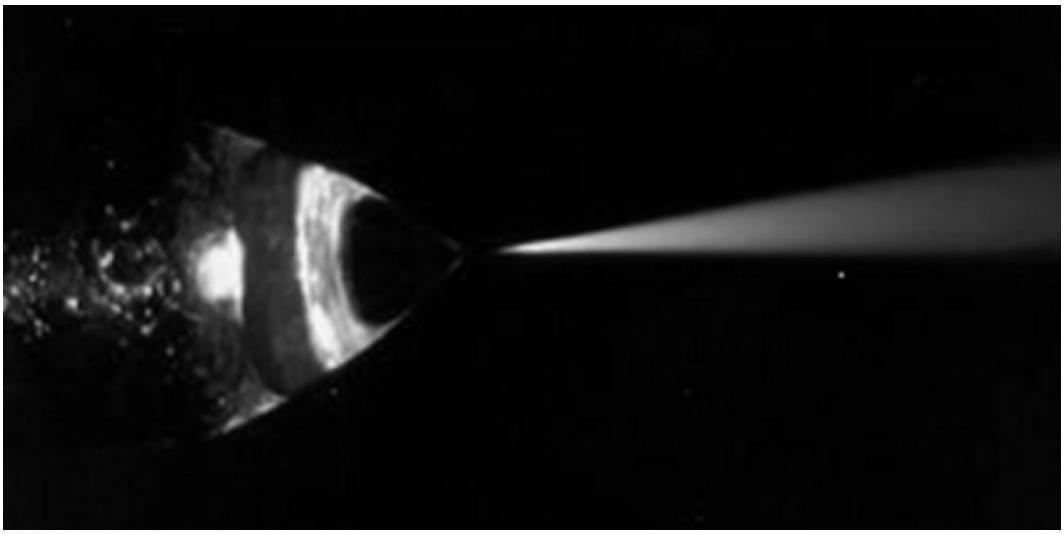


Figure 1: Electrospray in vacuum. From left to right: capillary emitter (approximately 1mm outer diameter), Taylor cone, and beam of droplets. Extracted from [5].

1.2 Scope

- Develop a routine to integrate the equation of motion of a large array of charged droplets representing typical conditions in electrospray thrusters under the effect of the space charge generated by themselves and the electric field generated by the electrodes. An algorithm based in the Verlet integrator will be used to integrate the equations of motion. The droplets are sequentially introduced in the computational domain.
- Simulate the interaction between two charged droplets of the same polarity , referred to as Coulomb collision, and compare the scattering angle obtained with the analytical solution that exists for a Coulomb collision. Such study will be used to define a Δt for the Verlet algorithm that gives an acceptable error.
- Upgrade the routine to compute the interaction of droplets with less computational cost. This will be done by dividing the domain in a cylindrical

grid and grouping droplets in neighborhoods shaped as rings. The electric field due to space charge acting on a droplet is then computed as the superposition of the electric field directly induced by droplets in its neighborhood, plus an averaged field induced by rings of charge for those droplets outside of the immediate neighborhood.

1.3 Requirements

- The experimental work must be done using an emitter/extractor/electrode geometry similar to that planned for LISA, as provided by JPL.
- The model must be validated with experimental work.
- The error committed while integrating the equation of motion with the Verlet algorithm must be inferior to 0.1%.
- Both the diode (emitter and extractor electrodes) and triode (emitter, extractor and accelerator electrodes) configurations must be considered.

1.4 Background

As the first dedicated space-based gravitational wave observatory, the Laser Interferometer Space Antenna (LISA) will detect waves generated by binaries within the Milky Way, and massive black holes in distant galaxies. To shield the gravitational wave instrument from disturbances that could mask the influence of gravitational waves, LISA consists of a precisely controlled set of spacecraft that follow the array proof masses within approximately 10 nm. The precision control of the spacecraft will be provided by microthrusters, and electrospray thrusters are a main technology candidate for this function.

A colloid thruster is made of several electrospray emitters (typically between 6 and 12). Each electrospray generates a beam of charged droplets, which are accelerated electrostatically in the region between the emitter and an extractor electrode. A thin jet emerges from the tip of the liquid meniscus and, upon its breakup (induced by capillary instability), generates charged droplets. Molecular ions are also

emitted from the surface of the jet and possibly from charged droplets. An orifice in the extractor permits each emitter's beam to exit to outer space. Under certain conditions the beam will impinge on the extractor, which is unfavorable from both performance and life points of view. The spreading of the beam occurs near the emitter tip, as a result of electrostatic forces induced by the beam's space charge and then farther downstream in the needle/extractor region as a result of the external electric field. Beam interception by the extractor and accelerator electrodes is a major risk for achieving the required thruster lifetime, as the progressive accumulation of fluid on the electrodes eventually leads to backspray between the grids, and ultimately to catastrophic shorting.

2 Problem description

2.1 Equations of motion of the droplets and dimensionless coefficients

The equations of motion of each particle is given by Newton 2nd Law. The force acting on each particle is the product of the electric field at the position of the particle times its charge. A good approximation for the electric field is a decomposition into two terms, induced by the space charge and the electrodes:

- **Space charge:** Electric field created by the surrounding droplets in the domain.
- **External electric field:** The electric potential difference between the extractor and the emitter produces an electric field.

The equation of motion in vectorial form for one droplet is described in Eq. (1).

$$m_i \frac{d^2 \vec{x_i}}{dt^2} = q_i \sum_{j=1}^{N(j \neq i)} \frac{q_j}{\epsilon_0 4\pi} \frac{\vec{x_i} - \vec{x_j}}{|\vec{x_i} - \vec{x_j}|^3} + q_i \, \vec{E}_{ext} \tag{1}$$

where:

m_i is the mass of the droplet.
$\vec{x_i}$ is the position vector of the droplet in Cartesian coordinates.
$\vec{x_j}$ is the position vector of the surrounding droplets in Cartesian coordinates.
q_i is the electric charge of the droplet.
q_j is the electric charge of the surrounding droplets.
ϵ_0 is the vacuum permittivity.
$\vec{E}_{ext}$ is the electirc field generated by the electrodes. N is the number of droplets in the domain.

In order to work more comfortably, the equation is written in dimensionless form using the following characteristic parameters:

- **Characteristic length:** $L_c = \lambda_j$

λ_j is the jet's wavelength associated with the average droplet. It is based on ξ_j and a droplet charged at 67.9% of its Rayleigh limit. The Rayleigh limit is the maximum charge that a droplet can hold before rupturing into daughter droplets.

- **Characteristic velocity:** $V_c = v_j$
  v_j is the velocity of the jet at the breakup point.

- **Characteristic time:** t_c
  The characteristic time is defined as:

$$t_c = \frac{L_c}{V_c} \tag{2}$$

  The characteristic time gives us an approximation of how long it takes for the droplet to advance a distance L_c given the initial velocity it has.

- **Characteristic charge to mass ratio:** $\xi_c = \xi_j$
  ξ_j is defined through the current of the beam and the flow rate of the dielectic liquid.

$$\xi_j = \frac{I}{\rho\ Q} \tag{3}$$

- **Characteristic charge:** $q_c = q_j$
  q_j is the charge of an average droplet.

$$q_j = \xi_j \rho \lambda_j \pi \frac{d_j^2}{4} \tag{4}$$

  where d_j is the diameter of the jet at the breakup point

$$d_j = \sqrt{\frac{4Q}{\pi v_j}} \tag{5}$$

Once the dimensionless parameters are introduced in Eq. (1), dimensionless coefficients appear: π_C and π_X.

Eq. (6) is the equation of motion in dimensionless form.

$$m_i \frac{d^2\vec{x_i}}{dt^2} = \pi_C \ \xi_i \sum_{j=1}^{N(j\neq i)} q_j \frac{\vec{x_i} - \vec{x_j}}{|\vec{x_i} - \vec{x_j}|^3} + \pi_X \ \xi_i \ \vec{E}_{ext} \tag{6}$$

where:

$$\pi_C = \frac{\xi_c \ t_c^2 \ q_c}{4\pi\epsilon_0 \ L_c^3} \tag{7}$$

$$\pi_X = \frac{E_c \ \xi_c \ t_c^2}{L_c} \tag{8}$$

Due to the fact that the model will focus on the space charge interaction, only the dimensionless coefficient related to the space charge interaction will be discussed.

Applying the above mentioned definitions of the dimensionless parameters to Eq. (7), it yields to Eq. (9).

$$\pi_C = \frac{I^2}{\rho \ Q \ v_j^3 \ 4\pi\epsilon_0} \tag{9}$$

π_C gives information about the ratio of potential energy to kinetic energy. It tells how large is the electric interaction compared to the kinetic energy of the droplet. π_C is obtained experimentally. All the variables are fairly easy to obtain except for the characteristic velocity. The obtention of such parameters is covered in Section 2.2.

To illustrate this behaviour an interaction between two droplets will be studied for different values of π_c. A Coulomb collision study will be used to reproduce the interaction of two close droplets. The main goal of such study is to derive a criterion for how small the time step in the Verlet algorithm needs to be in order to have a desired accuracy. See Section 3.3. Figure 2 illustrates a Coulomb collision. One particle is fixed and the other one goes from infinity to the position of the

fixed particle with an initial horizontal velocity. The moving particle is typically referred as "projectile" whereas the fixed particle is referred as "target". The initial separation between both particles is called the "impact parameter".

As a result of the interaction, the trajectory of the projectile is a hyperbola and the angle that it describes when it has gone beyond the position of the target is called the "scattering angle".

To illustrate how the interaction between the particles is affected by π_C, the scattering angle will be evaluated for different values of π_C. It is expected that for larger values of π_C the scattering angle will be larger because the interaction of both particles will be greater.

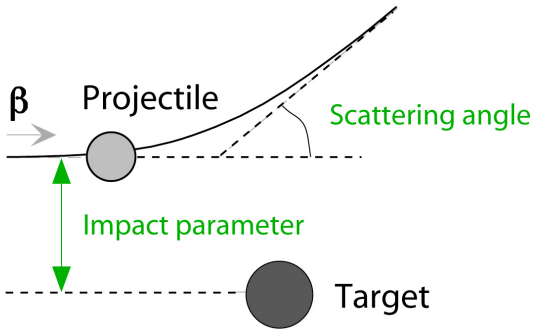


Figure 2: Illustration of a Coulomb collision. Extracted from

To compute the interaction between both particles the exact same functions used in the model have been used. The dimensionless mass of the fixed particle has been set to 1e25 so as to fix it in one position.

The results clearly illustrate the meaning of π_C. See Figure 3. As expected, the scattering angle increases with π_c.

For a very low value of π_c ($\pi_c = 0.01$), it can be seen that the trajectory of the projectile is nearly a straight line. In this case the kinetic energy of the projectile

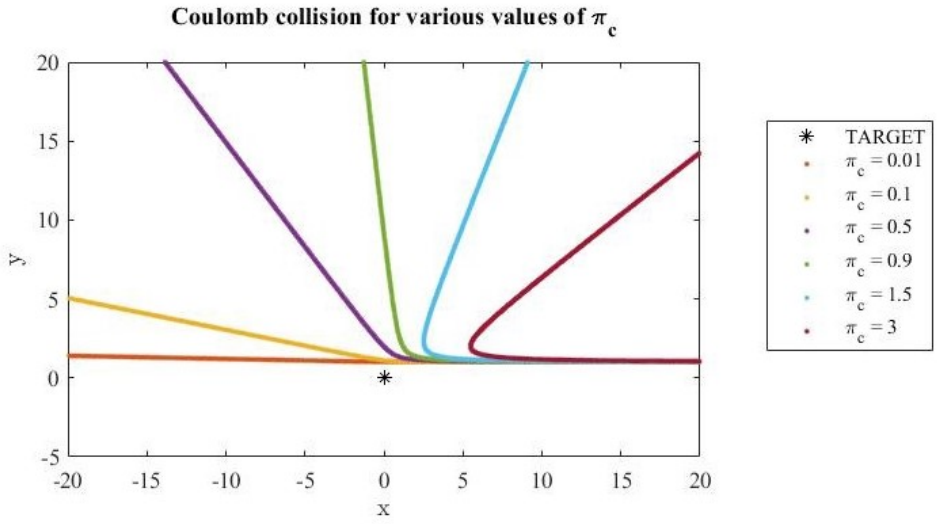


Figure 3: Coulomb collision for various values of π_c

is much larger than the electrical potential energy due to the interaction, thereby, the interaction is nearly negligible.
On the other hand, for a high value of π_c ($\pi_c = 3$), the projectile is really affected by the interaction with the target particle. The projectile completely changed its direction, its scattering angle is over 90^{o}.

2.2 Experimental input parameters and reproduction of the droplet population found in an electrospray beam

The integration of the equations of motion requires several input parameters to start the simulation. More specifically:

- The current of the beam (I), the flow rate (Q) of the dielectric liquid and the velocity of the jet (v_j) to evaluate the dimensionless parameter π_C.
- The position of the jet breakup point ($\vec{x}_{breakup}$).
- The distribution of the droplet's charge to mass ratio (ξ_i).
- The distribution of the droplet's charge (q_i) or alternatively the distribution of their associated jet wavelength (λ_i).

These parameters are obtained through time-of-flight and retarding potential measurements [5]. This section will cover the basic idea on how these methods are used to obtain the population of droplets that will be used in the model to recreate the structure of the beam.
Figure 4 shows an scheme of the system used [5].

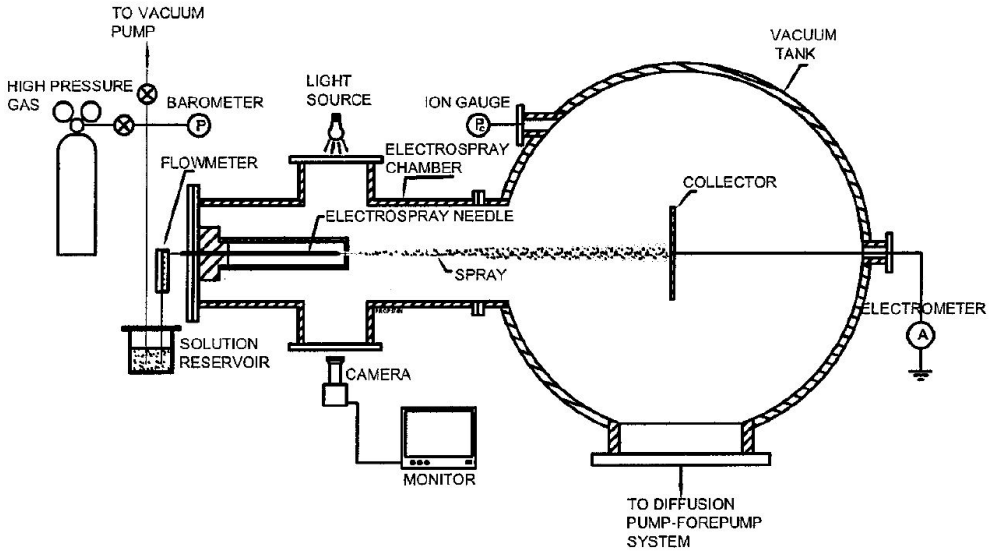


Figure 4: Schematic illustration of the hardware used to measure the initial parameters required for the model. Extracted from [5].

The retarding potential of a droplet ϕ_{RP}, defined as the sum of the kinetic and potential energy of the droplet divided by its charge (Eq. (10)) is a constant of the droplet in an electrostatic field [3].

$$\phi_{RP} = \frac{m}{2q} V(x)^2 + \phi \tag{10}$$

The retarding potential is measured by stopping the droplets with a screen and measuring the potential of it. Thus, the velocity is zero and therefore, the potential measured is the retarding potential.

Once the retarding potential is known, droplets' velocity is obtained through time-of-flight (TOF) measurements. The TOF is measured by means of an oscilloscope connected to the electrometer output and triggered by the voltage signal of the electrospray needle [5]. The TOF is the time it takes for a familiy of droplets to go from the emitter to the collector. Downstream the emitter the electric field is null, hence, the velocity of the droplets is constant. Given that the time it takes for the droplets to travel to the collector is known and the distance at which the collector is located is also known it is possible to obtain the velocity of the droplets downstream the extractor (Eq. (11)).

$$V = \frac{L}{t_{TOF}} \tag{11}$$

Finally, the specific charge ξ can be calculated with Eq. (10), it yields to Eq. (12).

$$\xi = \frac{q}{m} = \frac{L^2}{2\ \phi_{RP}\ t_{TOF}^2} \tag{12}$$

By moving the collector along the plane perpendicular to the axial direction of the beam it is possible to obtain the distribution of families of droplets and the current that they carry.

The model that we are developing in this thesis to reproduce the inner, space-charge dominated region of the beam needs a vector of droplets to be introduced sequentially in the domain. These droplets have to match the distribution of specific charge and current found experimentally.

To do so, a large pool of droplets is created with 30 different families of a discrete value of ξ each family. Each family contains a large group of droplets with different diameters. The value of the diameter is obtained with a normal Gaussian distribution. The charge of each droplet depends on its diameter, as Eq. (13) states. R is the radius of the droplet and ρ the density of the liquid.

$$q_i = \xi_i \ \rho \ \frac{4}{3} \ \pi \ R_i^3 \tag{13}$$

The wavelength associated with each droplet (λ_i) is calculated with the diameter, i.e. it comes from a normal Gaussian distribution. Throughout mass conservation it is possible to obtain λ_i, the mass of the jet associated with the wavelenght of the droplet must be equal to the mass of the droplet, see Eq. (14).

$$\lambda_i \ \pi \ R_{jet}^2 \ \rho = \frac{4}{3} \ \pi \ R_i^3 \ \rho \tag{14}$$

Notice that if Eq. (14) is introduced in Eq. 13 it yields to Eq. (4).

Finally, the vector with the droplets is obtained by picking randomly droplets from the pool. It is not completely random because the chosen droplets have to meet certain conditions.
Droplets chosen have to match the specific charge and current distribution found experimentally. To do so, for each family of droplets a different number of droplets is picked up. The current given by these droplets has to match the current found experimentally. The current is the electric charge (Q) transferred through the surface over a time (t). The electric charge transferred is the sum of the charge of every droplet of the family and the time is computed through the sum of the wavelengths associated to the droplets of the family and the velocity of the jet (v_j), see Eq. (15).

$$I = \frac{Q}{t} = \frac{\sum_{i=1}^{N} q_i}{\sum_{i=1}^{N} \lambda_i} v_j \tag{15}$$

If the current carried by the droplets does not match at least to 95% the current found experimentally, droplets are chosen again. The process is repeated until a group of droplets for every family of ξ matches the experimental output.

The only parameter left to evaluate experimentally is the velocity of the jet (v_j). The velocity of the jet is a crucial parameter for the model, the dimensionless

parameter π_C is strongly related to it, see Eq. (16).

$$\pi_C \sim v_j^{-3} \tag{16}$$

The velocity of the jet is also used to compute the population of droplets, and it is an important parameter for it.

The velocity of the jet will be estimated combining time-of-flight and retarding potential measurements with the output of the cone-jet model, it is not a straight-forward task to do. In the present work, it has been estimated from experience [5].

2.3 Coupling of inner and outer regions of the domain

In order to solve the problem appropriately, two regions are differentiated in the domain:

- **Inner region:**
  The inner region is located at the tip of the emitter and its purpose is to capture the correct physics during the jet break-up and the earliest stages of the spread of the beam. Initially, the Coulombic repulsion dominates the expansion of the electrospray beam. Because of it, the model for the inner region computes the trajectory of all droplets individually, thus, capturing exactly the interaction between droplets. This approach has a high computational cost and only allows to simulate a short time span. However, due to the importance of the Coulombic repulsion in the overall expansion of the beam, it has to be done this way. The model for the inner region is capable of reproducing very accurately the behaviour of droplets but it needs experimental parameters to compute it. The experimental parameters are not straight-forward to obtain and require several different experimental techniques. The obtention of such parameters is detialed in Section 2.2.

The model for the inner region will be discussed throughout the thesis.

- **Outter region:**
  Once the beam has spread enough, the space charge effect becomes less relevant because the distance between droplets has increased to the point that the interaction can be considered negligible. Downstream the extractor, the forces acting on a droplet are negligible, thereby, the trajectory of droplets becomes straight lines [6]. To solve the problem of the outter region, an Eulerian time-independent model is implemented. The model consists in grouping droplets in envelopes and integrate their trajectories in space instead of in time. Thus, it is possible to simulate larger time spans accurately. This model requires as input initial position and velocity of droplets. These parameters are the output of the inner region's model.

  This method for simulating the beam was developed and successfully demonstrated in a previous JPL-UCI contract by Prof. Gamero [6].

3 Numerical implementation

3.1 Structure of the code

This Section is devoted to explain the structure of the code. In Annex 7.1 a flow diagram of the code can be found. This section is thought to be read together with the flow diagram. Each block is identified with a letter to make it easier to locate an explanation in the flow diagram. The code written in MATLAB can be found in Annex 7.4.

A- Initialization

- The code cleans the workspace to make sure it is starting from scratch and no previous variables' values are stored.
- The code is set to work with more decimals than the option selected by default.
- Characteristic parameters are initialized and the dimensionless number (π) is computed.
- The computational parameters required to perform the simulation are initialized: number of time steps and Δt.
- The droplet population is initialized. The following characteristics of each droplet are loaded:
  - Charge of the droplet (Q)
  - Specific charge of the droplet (Q_s)
  - Initial time (time at which the droplet has to be introduced in the domain) (t_0)
  - Initial axial velocity (Vz_0)
  - Initial x position (x_0)
  - Initial y position (y_0)

– Initial axial position (z_0)

B- Grid initialization

The grid to group the droplets in cells is created. A more detailed explanation of the grid can be found in Section 3.4.

The code allows to:

- Choose the size of each axial division independently.
- Choose the number of rings for every axial division.
- Choose the thickness of each ring independently.

The grid has been coded in this way to enable the possibility to adjust the dimensions of each cell depending on their position in the domain. It may be interesting to have smaller cells next to the emitter and have larger cells downstream.

C- Compute total time

The total time is the amount of time that has been integrated so far. It is calculated through Eq. (17).

$$T = step \cdot \Delta t \tag{17}$$

D- Check if a new droplet should be introduced in the domain

In order to know if a new droplet should be introduced in the domain the total time is compared to the initial time of the droplet. If the total time is larger, then the droplet is introduced and its characteristics (charge, initial position, etc.) are loaded in the matrices that describe the domain. The vector that contains the information of the droplets is sorted by the initial time of the droplets, so the next droplet after the last one introduced in the domain is the one that is checked.

E- Compute new position

Here it starts the Verlet algorithm. Through equation Eq. (22) the new position of all droplets is computed.

F- Locate droplets in the grid

Once the new position for all droplets is computed, they are located in a cell depending on their position. Because the positions of the cells are already known, it is fairly easy to locate the droplets in their corresponding cell.
The result is a $2\ x\ Nd$ matrix where Nd is the number of droplets in the domain. The first row is the axial index of the cell in which the droplet is located and the second row is the radial index. Thus, it is possible to know where each droplet is.

At the end of the function it is checked that actually the droplet has been located in a existing cell. If it does not, a warning sign appears. The purpose of it is to ensure that the grid has been defined big enough to contain all the droplets during all the simulation.

G- Find which cells have droplets in it

Obviously, not all cells contain droplets, thereby, it must be known which cells contain droplets so in further calculations only cells with droplets will be taken into account.
To do so, all the repeated columns in the matrix with the indices of the container cell for each droplet are removed. Thus, we have a matrix with the indices of cells in which there are droplets.

H- Compute volumetric density of charge of each ring

The volumetric density of charge of each ring is computed through Eq. (27). The charge of all the droplets in the cell (ring) is divided by the volume of the ring.

The volumetric density of charge of all rings is also used to know which empty cells

become occupied. The newly computed density of charge is compared with the old one and if an empty cell becomes occupied, the indices of this cell are stored. This information will be used afterward to compute the geometric factors.
Also, the indices of the cells that change their charge status are stored, regardless of whether they were empty. This information will be used to only compute again the electric field if the density of charge of the cell has changed.

I- Compute geometric factors of cells that become occupied

The geometric factors are used to compute the electric field due to the rings at the nodes with a low computational cost. See Section 3.4.3 for a more detailed explanation.

Each cell has a geometric factor for every node in the grid. The geometric factor is decomposed in cylindrical coordinates, thereby, there are two types of geometric factors: one to compute the axial electric field and one to compute the radial electric field.

A N_A x N_R matrix has a slot for every node in the grid. N_A is the number of axial divisions of the grid and N_R is the number of radial divisions of the grid. In each slot (i.e. node) there are two matrices:

- A N_A x N_R matrix contains the **axial geometric factors** of all cells at this node.
- A N_A x N_R matrix contains the **radial geometric factors** of all cells at this node.

The geometric factors are independent of time, they only have to be computed once. In order to avoid computing more factors than needed, only when a cell becomes occupied the geometric factors of this cell are computed and stored.

J- Compute electric field at the nodes

The electric field at a node is computed through Eq. (50).

The key to improve the performance of the code is to avoid computing Eq. (50) for all cells at every time step. Instead, the electric field at the node is computed once and then updated with the new charge status of the cells.
More specifically, given that the cells that changed their charge status is known, the old contribution of these cells to the electric field is subtracted and the new one is added. Thus, it is not necessary to compute the contribution of all cells.

K- Find droplets in the influence zone

The model computes the electric field due to the space charge at each droplet position with a combination of two methods:

- The contribution of droplets that are far away is taken into account through the computation of the electric field due to the rings (groups of droplets) at the nodes.
- The contribution of droplets that are closer is taken into account in an exact manner, not an approximation. See Eq. (18). All variables in Eq. (18) are dimensionless.

$$\vec{E} = \sum_{i=1}^{N} q_i \frac{\vec{x_i} - \vec{x_0}}{|\vec{x_i} - \vec{x_0}|^3} \tag{18}$$

The distinction between "far droplets" and "close droplets" is done through the definition of an influence zone. The influence zone is defined as two axial divisions to each side of the axial division in which the droplet is found. However, it is possible to make it bigger or smaller.

Because the position of the axial divisions is known and the position of the droplets is also known it is easy to find which droplets are found in the influence zone.

An upgrade for the model will be to divide each ring in different cells and then compute the influence zone as the cells contained in an sphere centered in the cell in which the droplet is found. Thus, the volume of the influence zone would be significantly reduced and as a result of it, the computational time will also be reduced.

L- Compute electric field at the droplet position due to droplets in the influence zone

The electric field at the droplet's position due to all the droplets found in the influence zone is computed through Eq. (18). Because the list of droplets in the influence zone is already known it is possible to compute the electric field at the droplet's position.

If the contribution of all droplets had been done in this way, the computational time would be too large, making the study impossible. See Section 3.4 and Table 2.

M- Compute electric field at droplet's position due to the rings

So far, the electric field due to the rings of charge has been computed at the nodes of the grid, but not at the exact position of the droplet. To obtain the electric field at the exact position of the droplet a bilinear interpolation between the four corners of the cell is used. See Eq. (51). More detailed information about the bilinear interpolation is found in Section (3.4.3).

The electric field due to the rings is calculated in cylindrical coordinates (i.e. axial and radial coordinates). In order to later compute the acceleration of the droplet, the electric field must be in Cartesian coordinates (i.e. x, y and z coordinates).

N- Subtract electric field due to the rings found in the influence zone

The electric field computed at the nodes takes into account all droplets in the domain. In order to avoid counting double the contribution of the droplets found

in the influence zone, the electric field due to those rings found in the influence zone has to be subtracted.

The process is very simple because the geometric factors and the density of charge of these rings is already computed. The electric field due to these rings is computed again at the four nodes in which the droplet is located. Finally, a bilinear interpolation is applied to know the value of this electric field at the exact position of the droplet.

It is also necessary to express the electric field in Cartesian coordinates instead of cylindrical.

O- Compute final electric field at the droplet's position

Finally, the total electric field at the droplet's position is computed. See Eq. (19). The total electric field is computed by:

- Adding the contribution of the droplets in the influence zone. ($\vec{E}_{IZ}$)
- Adding the contribution of all rings. ($\vec{E}_{R}$)
- Subtracting the contribution of all rings contained in the influence zone. ($\vec{E}_{RIZ}$)

$$\vec{E} = \vec{E}_{IZ} + \vec{E}_{R} - \vec{E}_{RIZ} \tag{19}$$

P- Compute droplet's new acceleration and new velocity

The last step is to compute the new acceleration of the droplet and then the new velocity. The acceleration is computed dimensionlessly through Eq. (20).

$$\vec{a}_i = \pi_c \, \xi_i \, \vec{E}_i \tag{20}$$

- π_c is the dimensionless parameter described in Section 2.1.

- ξ_i is the specific charge of the droplet.
- $\vec{E_i}$ is the total electric field.

Once the new acceleration is computed, through Eq. (23), the new velocity is evaluated. This velocity will be used to compute the position of the droplet at the next step.

3.2 Verlet algorithm to solve the ordinary differential equations

Exact analytical solutions for the equations of motion exist only for very simple systems. It is not possible to obtain an analytical solution for our problem, therefore a numerical integrator is used to solve the set of differential equations.

Essentially, numerical integration of the equations of motion rely on the discretization of time. Acceleration, velocity, and position of particles are computed for every time step. The difference in time between time steps, hereby referred as Δt, is a crucial parameter for the algorithm. The Δt must be small enough to capture the motion of the particles but the smaller it is the more iterations needed for a specific time span. Thereby, the value of Δt is a trade-off between accuracy and computational cost. In models of explicit numeric integration there is a stability limit strictly related to the size of the time step used, such limit cannot be exceeded. The Δt used in our model will be justified in Section 3.3.

The Verlet algorithm was first introduced by Loup Verlet in 1967 [9] and it is the one used in our model to solve the movement of droplets in the beam. The original idea by Verlet considers a Taylor expansion of the position coordinate in two different directions of time:

$$x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{f(t)}{2m}\Delta t^2 + \frac{\partial^3 x}{3!\partial t^3}\Delta t^3 + O\left(\Delta t^4\right)$$

$$x(t-\Delta t) = x(t) - v(t)\Delta t + \frac{f(t)}{2m}\Delta t^2 - \frac{\partial^3 x}{3!\partial t^3}\Delta t^3 + O\left(\Delta t^4\right)$$

The symmetry in time found in the method reduces the local error [7]. When summing up the two expansions, the odd-degree terms disappear:

$$x(t+\Delta t) = 2x(t) - x(t-\Delta t) + \frac{f(t)}{2m}\Delta t^2 + O\left(\Delta t^4\right) \tag{21}$$

Notice that no velocities are needed to compute the position of the particle at the next time step. The algorithm implemented in our model is slightly different to the Verlet algorithm because we do want to know the velocities of the droplets. The algorithm implemented is called "velocity Verlet algorithm" [8]. It is mathematically equivalent to the Verlet alogrithm described above but explicitly incorporates the velocity:

$$x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2 \tag{22}$$

$$v(t+\Delta t) = v(t) + \frac{a(t)+a(t+\Delta t)}{2}\Delta t \tag{23}$$

The standard implemenation of the velocity Verlet algorithm is done as follows (Figure 5 illustrates the scheme of the algorithm):

1. Start from initial system's setup
2. Calculate new position

$$x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2$$

3. Calculate the intermediate velocity

$$v\left(t+\frac{1}{2}\Delta t\right) = v(t) + \frac{1}{2}a(t)\Delta t$$

4. Calculate the new acceleration $a(t+\Delta t)$ since now the new position of particles is known.
5. Calculate new velocity

$$v(t+\Delta t) = v\left(t+\frac{1}{2}\Delta t\right) + \frac{1}{2}a(t+\Delta t)\Delta t$$

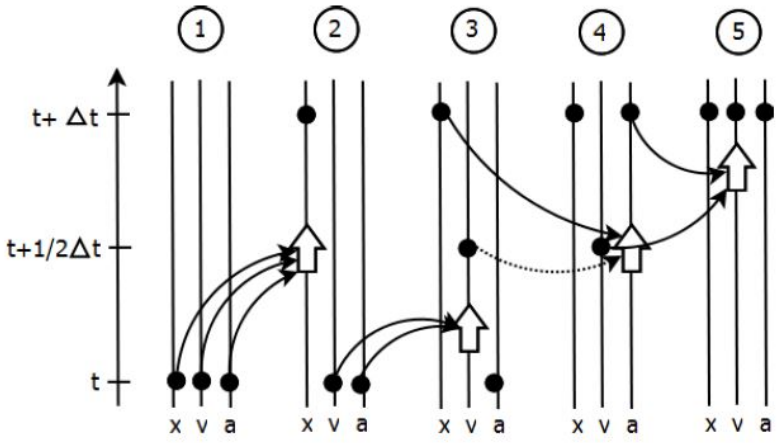


Figure 5: Illustration of the Velocity Verlet algorithm scheme. Extracted from [7]

The Verlet algorithm preserves physical properties of the system. Therefore, conservation of mechanical energy, linear and angular momentum should be fulfilled. We will use the conservation of mechanical energy to proof that the code works correctly. See Section 3.3.2.

3.3 Justification of the Δt used for the Verlet integration

3.3.1 Comparison with a six-stage, fifth-order, Runge-Kutta method

It has already been mentioned how important it is to choose appropriately the Δt used for the integration of the equations of motion. To justify the value used, the same scenario of a Coulomb Collision used in Section 2.1 to explain the sense of the dimensionless parameter π_C will be used.

A Coulomb collision is the most unfavorable case during the interaction because the distance between droplets is very small. To capture the effects and the interaction between droplets a Δt small enough must be used. If the Δt is too large, the interaction when droplets are very close will be inaccurate and it will affect the simulation afterward.

The sample of droplet population used for the simulation leads to a value of:

$$\pi_C = 0.0074 \tag{24}$$

The scattering angle for a Coulomb collision with $\pi_C = 0.0074$ will be evaluated through the Verlet algorithm used in the model and then it will be compared to the solution given by the built-in MATLAB function to solve ordinary differential equations.
MATLAB uses a six-stage, fifth-order, Runge-Kutta algorithm to solve ODE. This method is very accurate and can be considered as a reference solution [2].

The value of Δt should be inferior to 1. A Δt of 1 means that at every time step the particle moves approximately the value of the characteristic length (λ). See Section 2.1 for further information about the characteristic parameters. λ is the average distance of the jet breakup between droplets, having a larger value than 1 of Δt would mean that at some time steps more than one droplet would be introduced in the domain, which cannot happen.

The absolute error showed in Table 1 is computed as Eq. (25) states.

$$\text{error} = |\Theta_{\text{Runge-Kutta}} - \theta_{\text{Verlet}}| \tag{25}$$

	Scattering angle (Θ) (deg)		**Absolute error** (deg)
Δt	***Verlet***	***Runge-Kutta***	
0.05	0.847856	0.847870	**0.014052 e-3**
0.1	0.847860	0.847870	**0.009565 e-3**
0.3	0.847909	0.847870	**0.038928 e-3**
0.5	0.848030	0.847870	**0.160550 e-3**
0.7	0.848750	0.847870	**0.880129 e-3**

Table 1: Absolute error on the scattering angle computation for various values of Δt

All values of Δt lead to a very accurate solution. We require the model to have

an error less than 0.1 deg. Therefore, using a Δt of 0.1 is completely acceptable.

3.3.2 Conservation of mechanical energy

When only the space charge is acting on the droplets, energy must be conserved, therefore, the solution obtained with the Verlet algortihm must show a constant mechanical energy profile. If the Δt is not small enough, the integration of the equations of motion will be inaccurate and as a result of it mechanical energy will not be conserved.

In order to test if a Δt of 0.1 is small enough to fulfill energy conservation, 10 droplets have been forced to interact. Initially they have null velocity, so all the energy is potential energy. However, after some time, droplets have spread enough and all potential energy has been converted in kinetic energy. Figure 7 shows that at all time mechanical energy remains constant, indicating thus, that the integration of the equations of motion is correct and the value of Δt is enough. Figure 6 illustrates the motion of these 10 droplets.

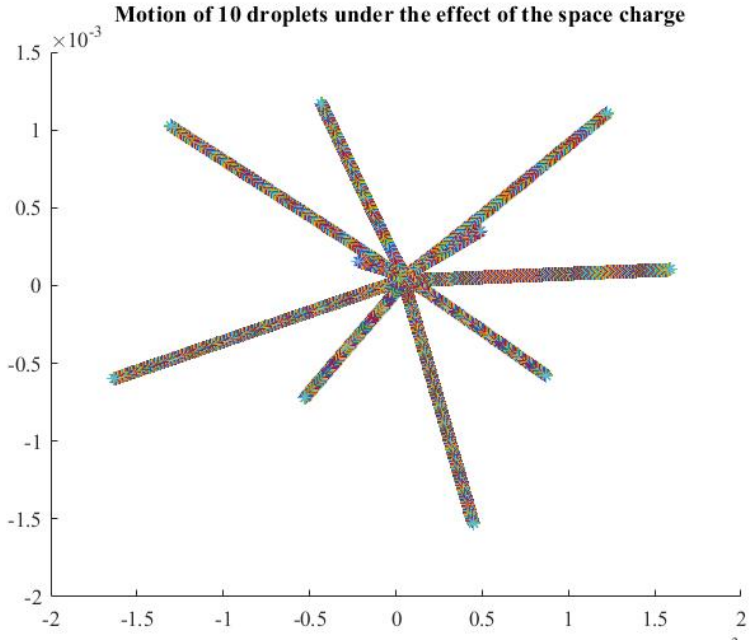


Figure 6: Motion of 10 droplets with null initial velocity

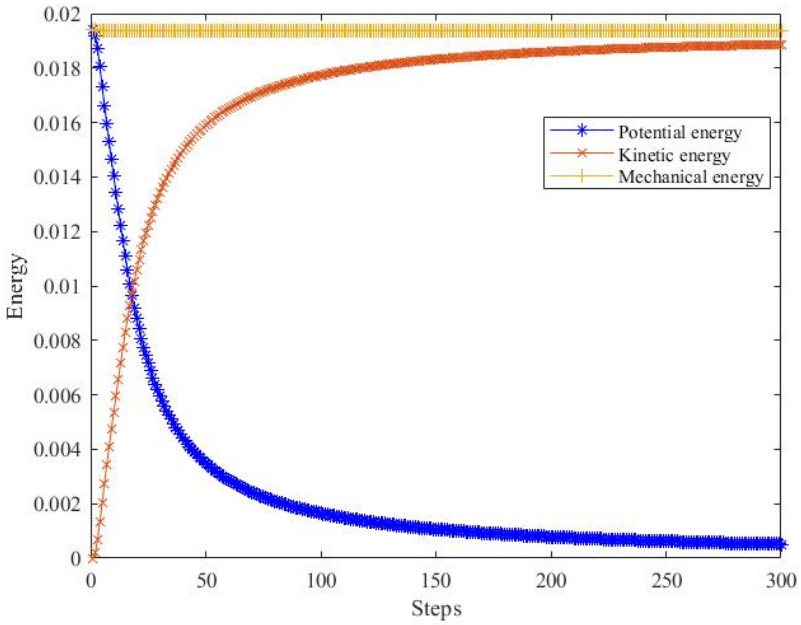


Figure 7: Energy of a system of 10 charged droplets.

In the model, when droplets are introduced sequentially one after another, energy is still conserved. When a droplet is introduced in the domain there is an increase in mechanical energy due to the initial velocity of the droplet. However, when all droplets are in the domain, mechanical energy remains constant. Figure 8 shows the energy balance of a system of 100 droplets introduced sequentially. The black marks on the plot indicate the time step in which a new droplet is introduced in the domain. In Figure 9 it can be clearly seen that mechanical energy is conserved at all times, only when droplets are introduced mechanical energy grows. Figure 8 has been obtained computing exactly the contribution of all droplets in the domain, the grid mentioned in Section 3.4 has not been used in this case. The grid enables the use of approximations to improve significantly the performance of the code, but because of these approximations, mechanical energy does not have to be exactly constant over time. In the best-case scenario it would be exactly constant.

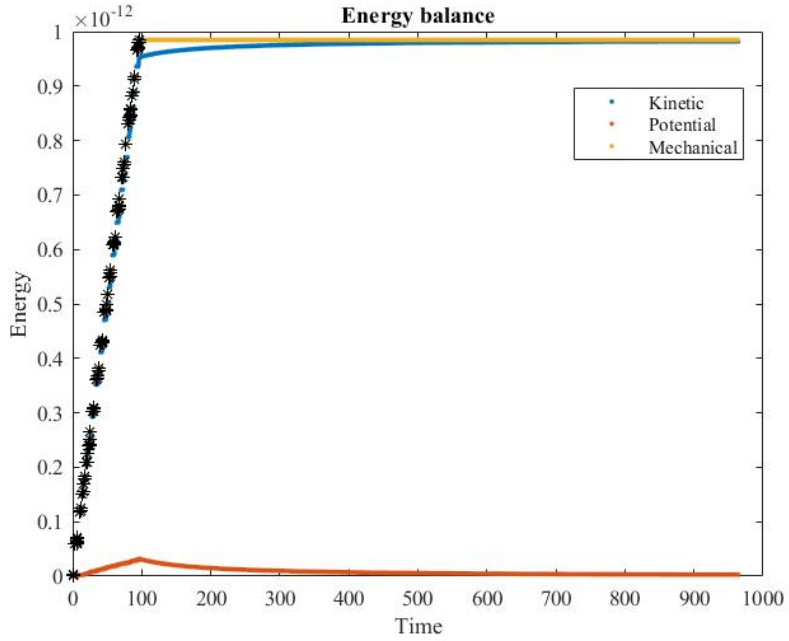


Figure 8: Energy balance of 100 droplets introduced sequentially in the domain.

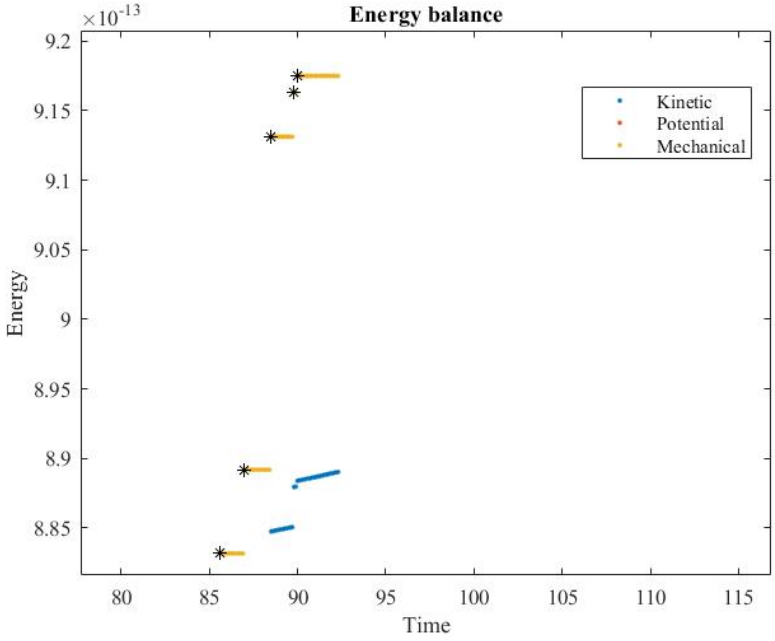


Figure 9: Amplified plot of Figure 8

3.4 Cylindrical grid to group droplets in ring cells

It is not possible to simulate the movement of large populations of droplets computing exactly the contribution to the electric field of all droplets in the domain (see Eq. (18)), the computational time would be too large. The number of iterations to compute the acceleration of particles at each time step grows as N^2. The electric field at each droplet position is found by adding up the contribution of all the droplets in the domain. Therefore, when a large number of droplets is introduced in the domain, the computational time increases to the point it is not possible to continue the study. Table 2 shows the elapsed time of the simulation for different number of droplets in the domain (simulated with MATLAB). Such information gives an idea of how is the growth of the computational time with the number of droplets.

Number of droplets in the domain	**Computational time**
20	0.102789 seconds
100	2.078435 seconds
200	15.643020 seconds
400	121.760476 seconds
10000	2140.410071 seconds.

Table 2: Computational time (MATLAB) for different number of droplets in the domain

To overcome the high computational cost mentioned above, a grid in cylindrical coordinates is introduced in the domain. Such grid will enable the possibility of doing an approximation and compute the electric field due to space charge using a ring of charge. The computation of the electric field generated by a ring of charge can be decomposed in the product of a geometric factor and the density of charge of the ring. The geometric factor will only be computed once, not at every time step, therefore, the computational time will be largely reduced. Detailed information about it is in Section 3.4.1.

The grid is created by dividing a cylinder that contains all the space in which the droplets will move during the simulation. The cylinder is divided N_A times

in its axial direction. The thickness of each axial division can be adjusted independently. Then, each axial division is divided N_R times into rings. The number of rings in each axial division and the thickness of the rings can also be adjusted independently. The rings will be frequently referred as cells.

3.4.1 Electric field of a ring of charge

The electric field generated by a ring of charge will be computed as the gradient of the electric potential, see Eq. (26).

$$E = -\nabla\psi \tag{26}$$

Even though rings do not have a uniform charge distribution, for the sake of simplicity, they are considered to be uniformly charged.
The volumetric density of charge (ρ) of the ring is computed as the quocient of the total charge contained in the ring and the volume of the ring, see Eq. (27).

$$\rho = \frac{\sum_{i=1}^{Nd} Q_i}{V_{\text{ring}}} \tag{27}$$

The total charge is computed as the sum of the charge of all the droplets contained in the ring. The density of charge is constant all over the ring, therefore the ring is charged uniformly.

Due to the fact that we consider the ring to have a uniform charge distribution we can compute the electric potential from a fundamental solution for Laplace's equation in three dimensions [1].
Eq. (28) is Laplace's equation expressed in terms of cylindrical curvilinear coordinates r, Ω and z for axisymmetric problems. ψ is the potential function.

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial r^2} + \frac{1}{r}\frac{\partial\psi}{\partial r} + \frac{\partial^2\psi}{\partial z^2} = 0 \tag{28}$$

Figure 10 illustrates an axisymmetric solution domain of arbitrary crosssectional

shape. Point "p" is a "load" point with coordinates R_p, Ω_p, z_p. In our case, the load point will be the point at which we will evaluate the electric field. Notice that the point is not fixed in the domain. Point Q, located in the boundary with coordinates R_Q, Ω_Q, z_Q, refers to a typical "field" point. In our case, we will have a distribution of field points that will form a ring over the boundary.

Eq. (29) describes the potential distribution generated when a load point is in the domain at a distance of $r(p, Q)$ of the boundary.

$$\psi' = \frac{1}{4\pi\ r(p, Q)} \tag{29}$$

A.A. Bakr expresses the distance between the load point and the boundary in cylindrical curvilinear coordinates so it is possible to integrate Eq. (29) in a circular path about the axis of rotational symmetry. The process yelds to Eq. (30).
The detailed steps followed to arrive at Eq. (30) can be found at [1].

$$\psi(p, Q) = \left(\frac{-1}{2\pi^2}\right) \frac{m}{(R_p\ R_Q)^{\frac{1}{2}}} K\left(m, \frac{\pi}{2}\right) \tag{30}$$

$K\left(m, \frac{\pi}{2}\right)$ (Eq. (31)) is the complete elliptic integral of the first kind of modulus m (Eq. (32)) and argument $\pi/2$.

$$K\left(m, \frac{\pi}{2}\right) = \int_0^{\pi/2} \left(1 - m^2 \sin^2 \alpha\right)^{-\frac{1}{2}} d\alpha \tag{31}$$

$$m = \frac{2\,(R_p\ R_Q)^{\frac{1}{2}}}{\left[(R_p + R_Q)^2 + (z_p - z_Q)^2\right]^{\frac{1}{2}}} \tag{32}$$

Eq. (31) is computed through an approximation. Evaluating exactly the integral

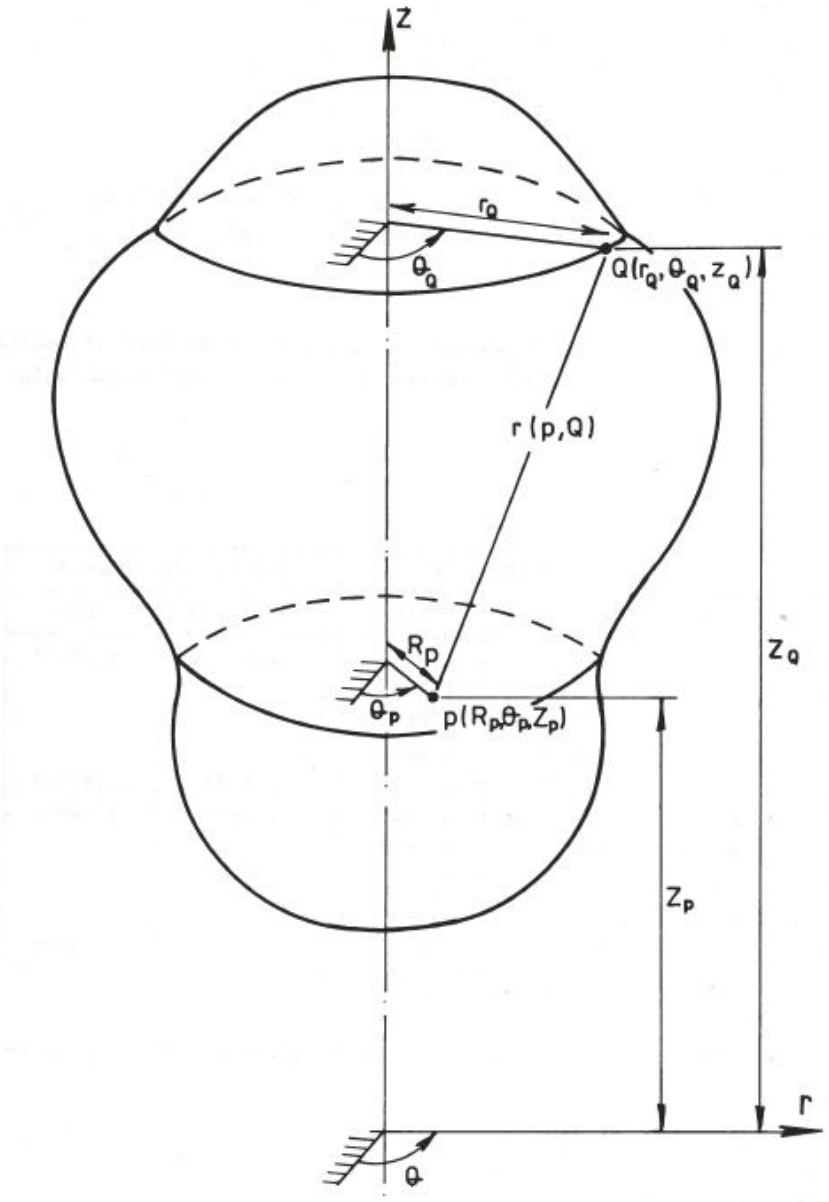


Figure 10: Typical axisymmetric solution domain

every time would significantly affect the performance of the code.

$$K\left(m, \frac{\pi}{2}\right) = \ln 4 + \sum_{i=1}^{5} A_i (1-m)^i + \ln\left(\frac{1}{1-m}\right)\left(0.5 + \sum_{i=1}^{5} B_i (1-m)^i\right) \quad (33)$$

The coefficients needed for Eq. (33) are:

- $A = [0.0965786196, 0.0315594316, 0.0237612248, 0.0259628884, 0.00663980111]$
- $B = [0.1249992959, 0.0701487577, 0.0449838755, 0.0187516602, 0.00184723416]$

Figure 11 shows clearly that the approximation is solid. $K\left(m, \frac{\pi}{2}\right)$ has been evaluated with the highest accuracy possible with the built-in MATLAB function and has been compared with the approximation described in Eq. (33).

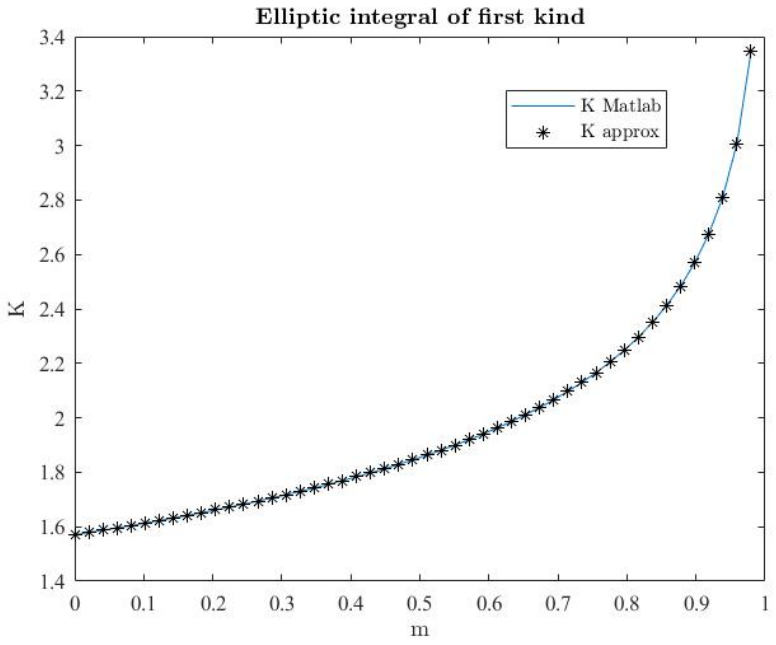


Figure 11: Elliptic integral of the first kind

Finally, to obtain the potential in the electrical form the multiplying constant $\left(\frac{-1}{2\pi^2}\right)$ from Eq. (30) must be adjusted. If the potential function is expressed in electrical form (Eq. (34)) it yields to a new expression of the potential distribution slightly different from Eq. (30). It adds the vacuum permitivity. Notice that the

expression of m has also been introduced. See Eq. (38).

$$\psi' = \frac{1}{4\pi\ \epsilon_0\ r(p,Q)} \tag{34}$$

$$\psi(p,Q) = \left(\frac{-1}{\pi^2\ \epsilon_0}\right) \frac{K\left(m, \frac{\pi}{2}\right)}{\left[(R_p + R_Q)^2 + (z_p - z_Q)^2\right]^{\frac{1}{2}}} \tag{35}$$

However, Eq. (38) will be used in its dimensionless form. In Eq. (36) all variables are dimensionless.

$$\psi(p,Q) = \left(\frac{-1}{\pi}\right) \frac{K\left(m, \frac{\pi}{2}\right)}{\left[(R_p + R_Q)^2 + (z_p - z_Q)^2\right]^{\frac{1}{2}}} \tag{36}$$

Once the electric potential is found it is possible to find the electric field through Eq. (26).

$$\vec{E} = -\left(\frac{\partial\psi}{\partial R_p}, \frac{\partial\psi}{\partial z_p}\right) \tag{37}$$

To do the partial derivatives of Eq. (37) I will express Eq. (38) as the product of two functions and a constant:

$$\psi(p,Q) = \left(\frac{-1}{\pi^2\ \epsilon_0}\right)\ f(R_p, z_p)\ K\left(m, \frac{\pi}{2}\right) \tag{38}$$

where:

$$f(R_p, z_p) = \frac{1}{\left[(R_p + R_Q)^2 + (z_p - z_Q)^2\right]^{\frac{1}{2}}} \tag{39}$$

Then, the partial derivatives of Eq. (37) are:

$$\frac{\partial\psi}{\partial R_p} = \left(\frac{-1}{\pi^2\ \epsilon_0}\right)\ \left[\frac{\partial f(R_p, z_p)}{\partial R_p}\ K\left(m, \frac{\pi}{2}\right) + f(R_p, z_p)\ \frac{\partial K\left(m, \frac{\pi}{2}\right)}{\partial m}\ \frac{\partial m}{\partial R_p}\right] \tag{40}$$

$$\frac{\partial \psi}{\partial z_p} = \left(\frac{-1}{\pi^2\ \epsilon_0}\right)\ \left[\frac{\partial f(R_p, z_p)}{\partial z_p}\ K\left(m, \frac{\pi}{2}\right) + f(R_p, z_p)\ \frac{\partial K\left(m, \frac{\pi}{2}\right)}{\partial m}\ \frac{\partial m}{\partial z_p}\right] \tag{41}$$

All partial derivatives in Eq. (40) and Eq. (41) are computed with MATLAB Symbolic toolbox except $\frac{\partial K\left(m, \frac{\pi}{2}\right)}{\partial m}$, which is computed manually. It yields to Eq. (42).

$$\frac{\partial K\left(m, \frac{\pi}{2}\right)}{\partial m} = \sum_{i=1}^{5} A_i(-i)(1-m)^{i-1} + \frac{1}{1-m}\left(0.5 + \sum_{i=1}^{5} B_i(1-m)^i\right) + \ln\left(\frac{1}{1-m}\right)\left(\sum_{i=1}^{5} B_i(-i)(1-m)^{i-1}\right) \tag{42}$$

In order to validate the code, a uniformly charged ring of charge has been reproduced with a perfect distribution of droplets. Figure 12 shows a ring of charge formed by 100 droplets. The electric field at an arbitrary point ($\vec{x_0}$) will be computed summing up the contribution of each droplet in the ring, see Eq. (43), and then compared to the electric field obtained through Eq. (37).

$$\vec{E} = \frac{1}{4\pi\epsilon_0}\sum_{i=1}^{N} q_i \frac{\vec{x_i} - \vec{x_0}}{|\vec{x_i} - \vec{x_0}|^3} \tag{43}$$

The same can be done for the electric potential, it can be computed summing up the contribution of each droplet in the ring, see Eq. (44), and then compared to the electric potential obtained through Eq. (38).

Finally, if the point of observation is located at the axis of the ring, the previous two solutions can be also compared with the analytical solution of the electric field on the axis of a ring of charge.

All methods have lead to the same values. Therefore, it can be confirmed that the code works properly. Also, our results agree with results from Zypman, Fredy R [11], who computes the electric field off-axis following a complete different method.

$$\psi = \sum_{i=1}^{N} q_i \frac{\vec{x_i} - \vec{x_0}}{|\vec{x_i} - \vec{x_0}|^2} \tag{44}$$

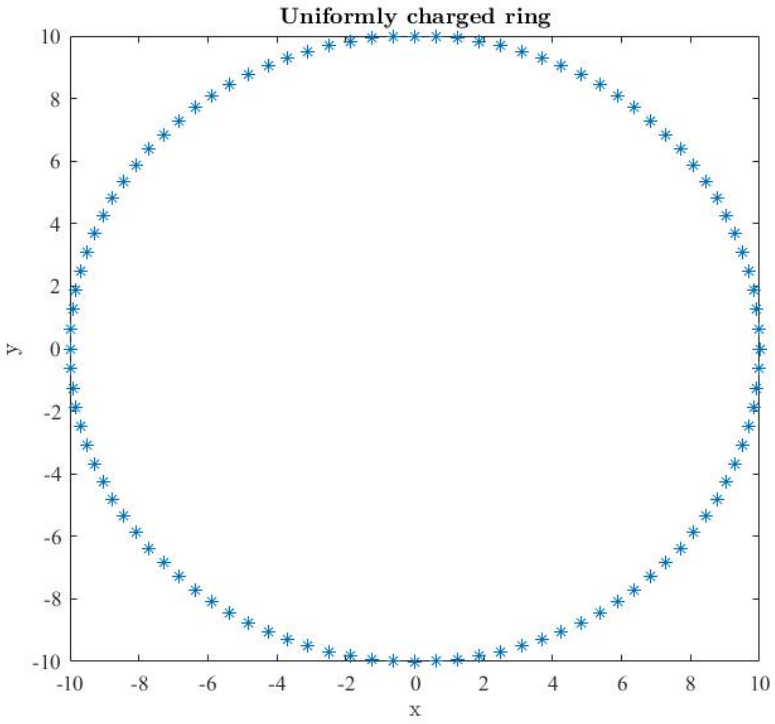


Figure 12: Uniformly charged ring

3.4.2 Integration of the electric field of a ring over a cell using Gauss quadrature method

The electric field generated by a ring of charged described in Section 3.4.1 accounts only for a 1D ring whereas the cells of the grid are 3D rings. Therefore, Eq. (37) must be integrated in the full domain of the cell. See Eq. (45). To do so, a 4 point Gaussian quadrature rule is applied two times: one to integrate in the radial

direction and one to integrate in the axial direction.

$$\vec{E} = \rho \int_{z_1}^{z_2} \int_{r_1}^{r_2} (-\nabla\psi) dr dz \tag{45}$$

The Gaussian quadrature rule allows to compute an integral by summing up the function evaluated at specific points multiplied by a specific factor, typically referred as "weights". See Eq. (46). The points at which the function is evaluated and the weight of each point depends on the number of points used to do the integration. The position and weight for a 4 point Gaussian quadrature rule are listed in Table 3.

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i) \tag{46}$$

Point	**Point position,** x_i	**Point weight,** w_i
1	-0.861136	0.347855
2	-0.339981	0.652145
3	0.339981	0.652145
4	0.861136	0.347855

Table 3: Position and weight of the four points needed for the Gaussian quadrature

However, now Eq. (46) has to be adapted to match our problem. The limits of integration of Eq. (45) are not -1 and 1. Therefore, a change of variable must be applied to bring Eq. (45) to a domain where the limits of integration are -1 and 1, hence, allowing the use of the Gaussian quadrature rule. After applying the change of variable, the resulting equation is Eq. (47).

$$\int_a^b f(x)dx = \int_{-1}^{1} f\left(\frac{b-a}{2}x + \frac{b+a}{2}\right)\frac{b-a}{2}dx \tag{47}$$

3.4.3 Computing the electric field at the nodes of the grid

The final step to improve the performance of the code is to compute the electric field due to the rings at the nodes of the grid instead of computing it at the exact position of every droplet. Then, the value of the electric field due to the rings at the exact position of the droplet is evaluated doing a bilinear interpolation between the corners of the ring.

The electric field due to a ring can be computed as a product of a geometric factor and the density of charge of the ring (Eq. (48)). If the electric field is computed at the nodes instead of at the exact position of each droplet, the geometric factor does not change. Because it does not change, it is possible to compute the geometric factor only once. Then, only the density of charge has to be computed at every time step.

$$\vec{E}_i = \rho_i \ \vec{R}_i \tag{48}$$

- **Geometric factor:** It is independent of time. It only depends on the dimensions of the ring and the position of the node. It is only computed once. It is computed through Eq. (49).

$$R_i = \int_{z_1}^{z_2} \int_{r_1}^{r_2} (-\nabla\psi) dr dz \tag{49}$$

- **Density of charge:** It is dependent of time. It depends on the droplets contained by the ring. It is computed every time step. It is computed through Eq. (27).

The electric field at a node is found by adding up the contribution of all rings in the domain. See Eq. (50).

$$\vec{E}_{node} = \sum_{i=1}^{Ncells} \rho_i R_i \tag{50}$$

Figure 13 illustrates the sense of the bilinear interpolation. Eq. (51) provides the value of the function at (x, y) interpolating from the value of the function at the

corners.

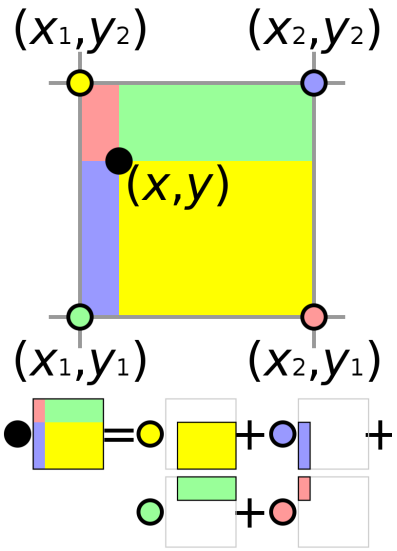


Figure 13: Illustration of a bilinear interpolation

$$f(x,y) \approx \frac{1}{(x_2-x_1)(y_2-y_1)} \begin{bmatrix} x_2-x & x-x_1 \end{bmatrix} \begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} y_2-y \\ y-y_1 \end{bmatrix} \tag{51}$$

4 Results

This Section is devoted to explain the results obtained with the mathematical model explained throughout this thesis. It only covers the results extracted with the model of the inner region. The results of the complete model, inner region coupled with outer region, are not complete at the moment being. The results have been obtained with a code written in C to boost the performance.

4.1 Droplet sampling

The droplet population has been divided into 32 groups of fixed specific charge (ξ). The total population consists in 70.005 droplets, the current of the beam is 400 nA and the characteristic values used for the dimensionless equations of motion are:

- **Characteristic specific charge (ξ_c):** 549.24 C/kg
- **Characteristic charge (q_c):** $2.57329x10^{-17}$ C
- **Characteristic length (λ_c):** $4.1387x10^{-8}$ m
- **Characteristic velocity (V_c):** 643.3 m/s

Figure 14 shows the specific charge distribution for the droplet population used in the simulation. The y axis displays the accumulated current of the beam; not all families have the same contribution to the overall current. The higher the step is, the more charge is carried by that specific family of ξ. It can be seen that families of droplets with a large specific charge carry less charge than the families with a smaller specific charge. Notice that the scale of the ξ axis is logarithmic. Figure 15 is not in logarithmic scale and it is easier to observe the contribution of each family to the overall current.

Both Figures 14 and 15 end up with an accumulated current of 1. It means that the contribution of all families to the overall current has been taken into account, thereby, the current carried by the droplets is 100% of 400 nA, which is the total

current of the beam.

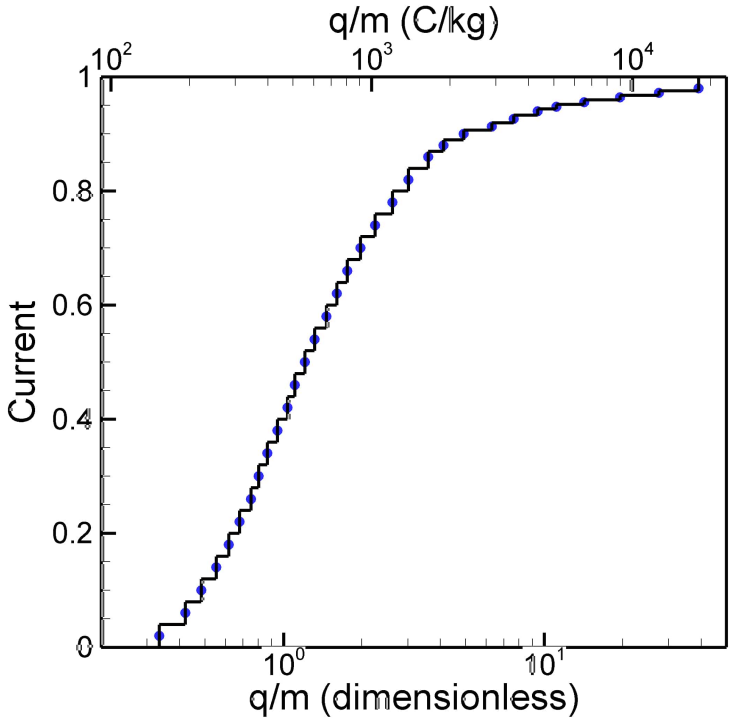


Figure 14: Specific charge (ξ) distribution

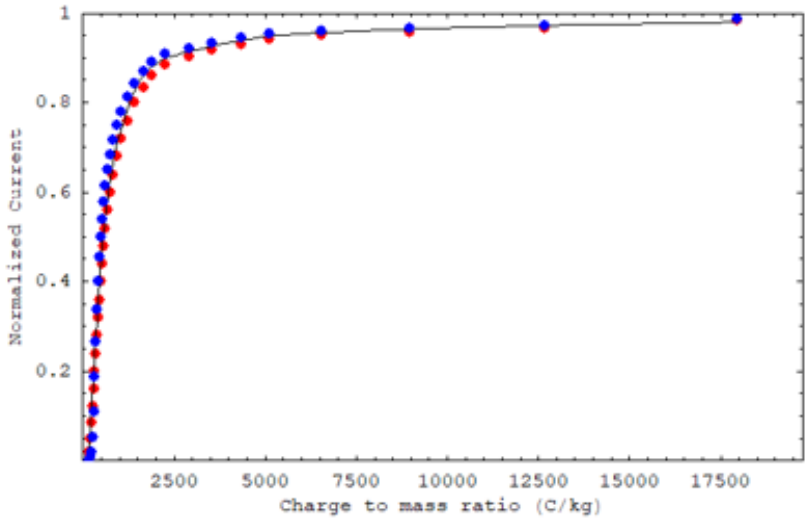


Figure 15: In **blue**: simulated ξ distribution
In **red**: experimental ξ distribution

As it has been explained in Section 2.2, the simulated ξ distribution matches the experimental ξ distribution. Figure 15 shows that it indeed matches.

4.2 Motion of droplets in the inner region of the domain

In the thesis defense presentation I will show a full movie of the motion of the droplets in the domain. Here, only some frames of this movie will be presented.

Notice that the time of each frame is dimensionless, it must be multiplied by the characteristic time to obtain the real time of the frame. The characteristic time, as it has been mentioned in Section 2.1, it is computed through the characteristic length and characteristic velocity of the problem. See Eq. (2). Such values are indicated above and they yield to a characteristic time of:

$$t_c = 6.43354x10^{-11}s \tag{52}$$

The following figures show the spreading of the beam. Droplets are stopped at $Z_{dimensionless} = 100$ because at this point is where the outter region starts. Notice how small the inner region of the domain is. The last frame, Figure 21, is the moment in which all droplets (70.005) are for first time in the domain. Notice, also, how short is the duration of the simulation. All the droplets are introduced in 32 ns approximately.

The straight lines displayed in the figures are the limits of the grid. Each square is a ring (cell). For this simulation a uniform grid has been used.

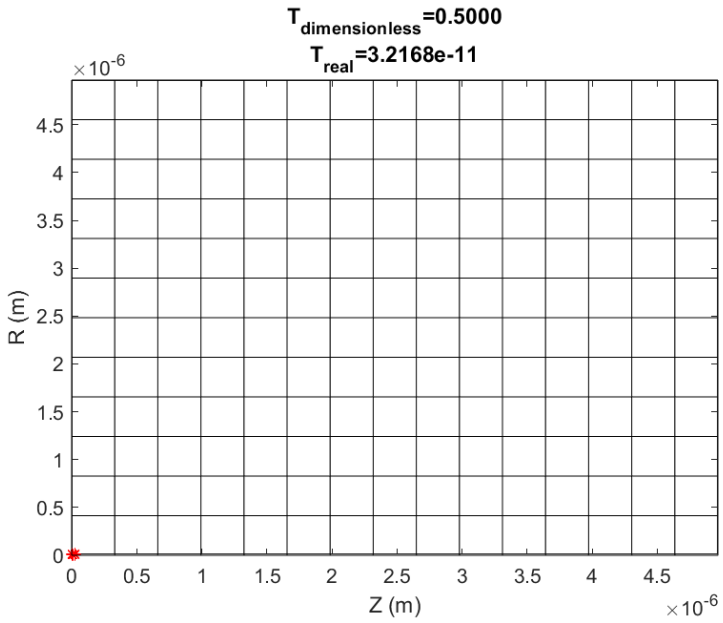


Figure 16: Motion of droplets in the domain at $T_d = 0.5$

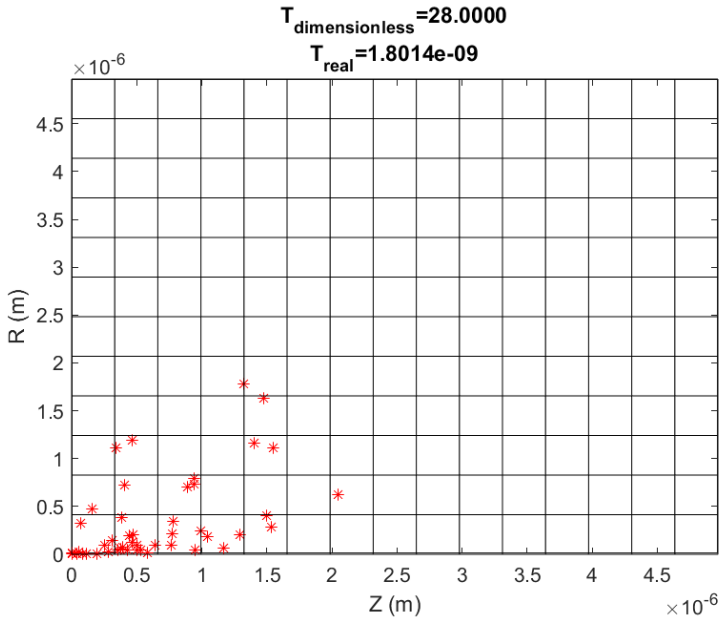


Figure 17: Motion of droplets in the domain at $T_d = 28$

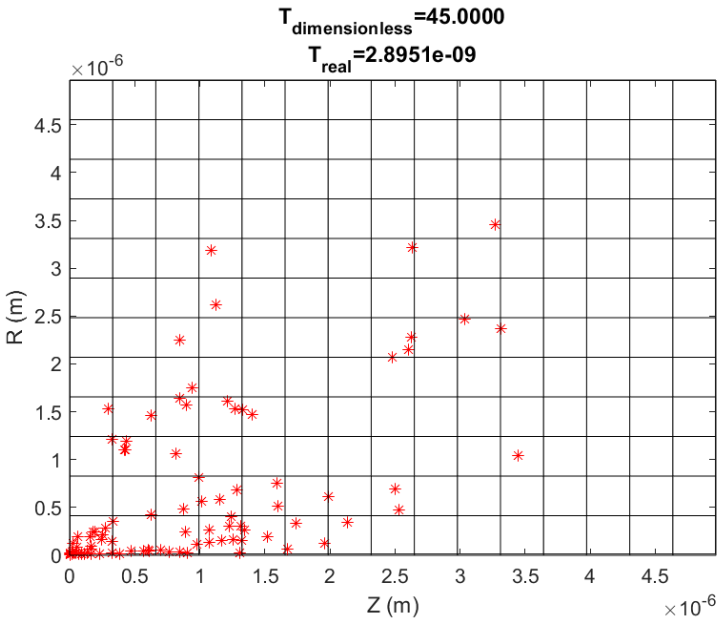


Figure 18: Motion of droplets in the domain at $T_d = 45$

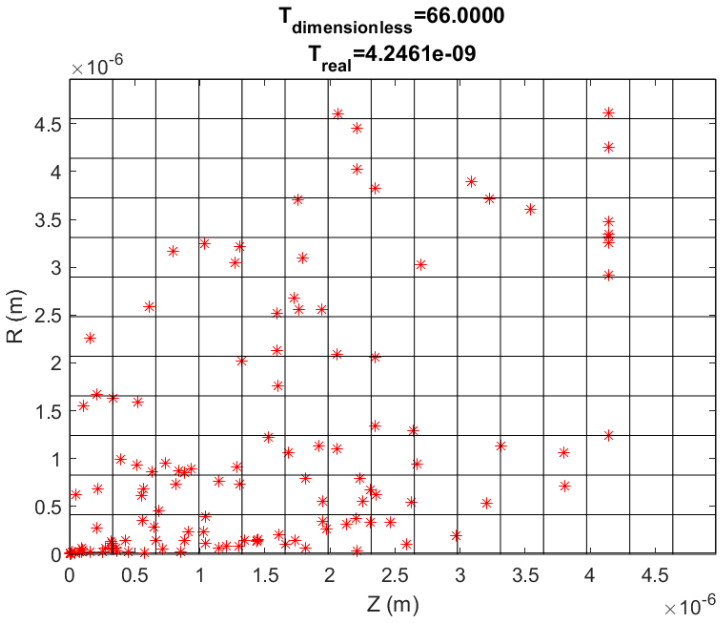


Figure 19: Motion of droplets in the domain at $T_d = 66$

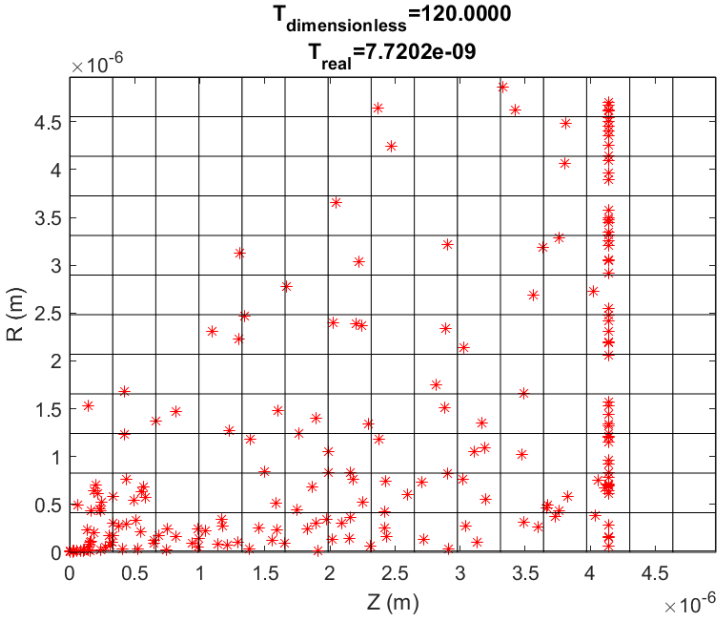


Figure 20: Motion of droplets in the domain at $T_d = 120$

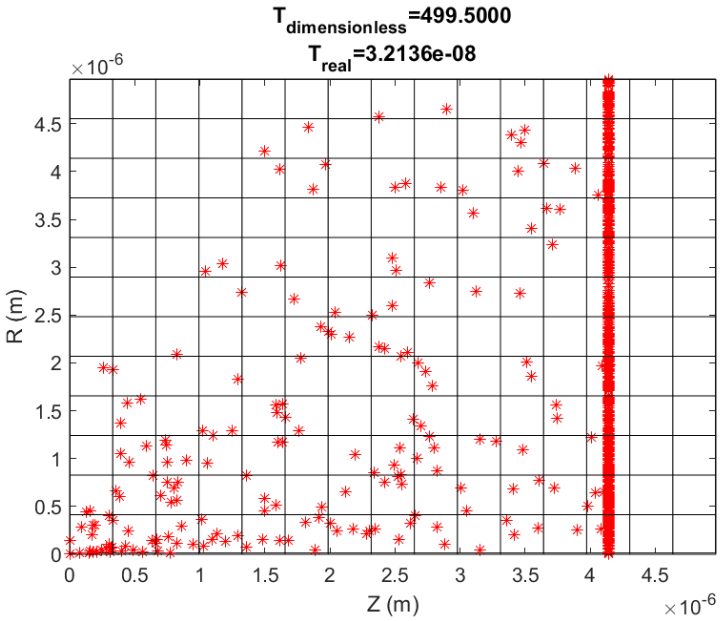


Figure 21: Motion of droplets in the domain at $T_d = 499.5$

4.3 Beam profiles

Figures 22 shows the overall beam profile and 23 shows the beam profile for different families of ξ. These profiles have been computed at the stopping point ($Z_{dimensionless} = 100$) and the polar angle is the angle that the straight trajectory of the droplet forms with the axis of the cylinder. These Figures show us how straight actually the beam is. In Figure 22 it is clearly seen that most of the current comes from droplets that are very close to the axis, but if we take a look at Figure 23 it can be seen that the distribution changes depending on the value of ξ of each family: smallest values still have the same overall tendency and most of the current is very close to the axis. However, if the specific charge is increased this behaviour changes, most of the current is between 10^{o} and 25^{o}. Finally, the family with the largest specific charge value has a very little contribution. Bear in mind that there are only a few droplets with such a large specific charge (see Figure 15) that's why the current density is so small.

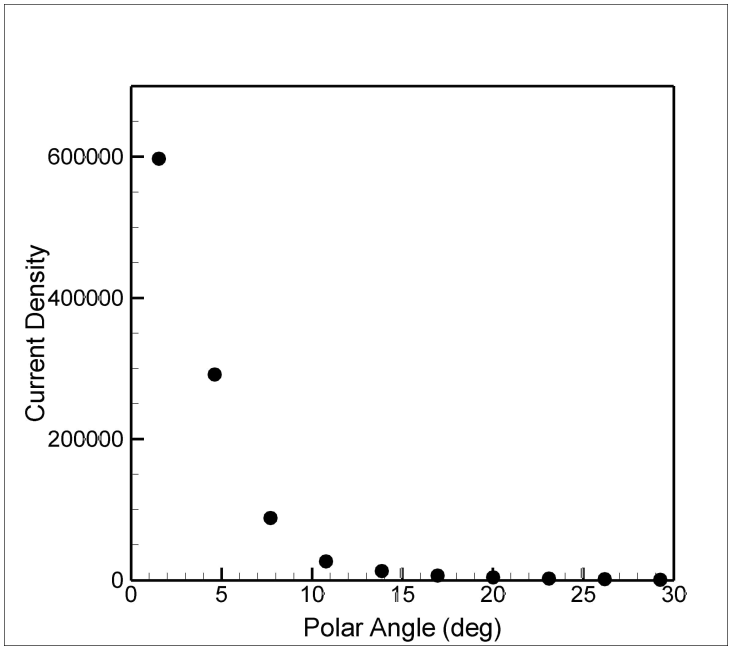


Figure 22: Overall beam profile at z=100

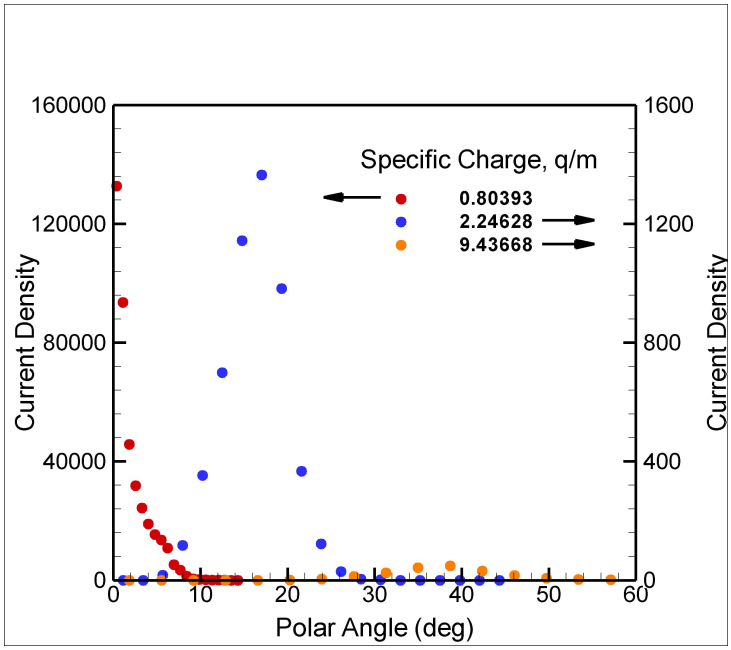


Figure 23: Beam profile for different ξ families at z=100

4.4 Position and velocity at the interface

The model for the outer region of the domain needs the position and velocity of the droplets at the interface (transition point between the inner region and the outer region). Figure 24 shows the velocity and position of groups of droplets of two different families of ξ. It can be seen that families with a larger value of specific charge tend to spread more than those with a smaller value of ξ. Not only this behaviour is seen in the position on the droplets but also on its radial velocity. Droplets from families with a large value of ξ tend to have greater radial velocities, so they will quickly spread downstream.

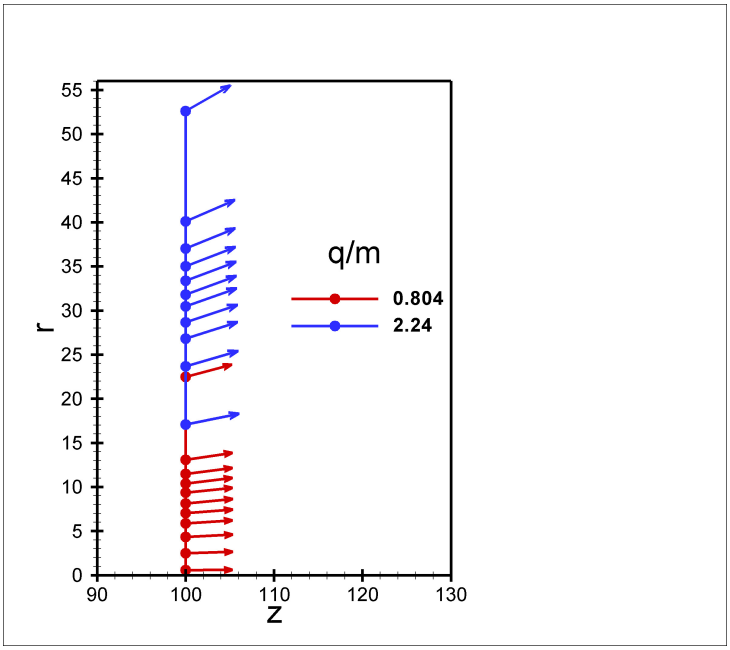


Figure 24: Position and velocity of groups of droplets at the interface point

5 Conclusions

The model for the inner region integrates successfully the motion of the droplets and captures the strong Coulomb interaction that takes place at the earliest stages of the beam. The approach used in the model is solid: the space charge effect is well approximated by the electric field generated by rings of charge and the computational cost has been reduced in a great extend by the usage of the grid. However, the key of the model is the computation of the geometric factors only one time and the obtention of the electric field of the rings just by updating it at every time step, not computing it from scratch.

Results show that the motion of the droplets is strongly related to their specific charge. Droplets with higher values of ξ tend to spread more easily. Further conclusions will be extracted when the full model of the beam is complete.

6 Future work

The next step will be to couple the model of the inner region with the model for the outer region. Once the full model is accomplished, it will be possible to validate the results experimentally. Furthermore, it will also be possible to obtain a value for the characteristic velocity by contrasting the output of the model with the output of the experiments. Eventually the model will also be upgraded to take into account the effect caused by ions. It has been observed experimentally that ions are also emitted from the beam, not only charged droplets, therefore, a more accurate model will also include the presence of ions.

Finally, once the model is fully validated and demonstrated, it will be used to evaluate easily how the beam behaves in different conditions.

References

[1] Adib A Bakr. *The Boundary Integral Equatio Method in Axisymmetric Stress Analysis Problems*. Vol. 14. Springer Science & Business Media, 2013, pp. 7–10.

[2] TA Elgohary et al. 'A simple, fast, and accurate time-integrator for strongly nonlinear dynamical systems'. In: *CMES: Computer Modeling in Engineering & Sciences* 100.3 (2014), pp. 249–275.

[3] CL Enloe and JR Shell. 'Optimizing the energy resolution of planar retarding potential analyzers'. In: *Review of scientific instruments* 63.2 (1992), pp. 1788–1791.

[4] William M Folkner et al. 'Disturbance reduction system: testing technology for precision formation control'. In: *High-Contrast Imaging for Exo-Planet Detection*. Vol. 4860. International Society for Optics and Photonics. 2003, pp. 221–228.

[5] M Gamero-Castano and V Hruby. 'Electrospray as a source of nanoparticles for efficient colloid thrusters'. In: *Journal of Propulsion and Power* 17.5 (2001), pp. 977–987.

[6] Manuel Gamero-Castaño. 'The structure of electrospray beams in vacuum'. In: *Journal of Fluid Mechanics* 604 (2008), pp. 339–368.

[7] Christian Holm. 'Simulation Methods in Physics'. In: ().

[8] William C Swope et al. 'A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters'. In: *The Journal of Chemical Physics* 76.1 (1982), pp. 637–649.

[9] Loup Verlet. 'Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules'. In: *Phys. Rev.* 159 (1 July 1967), pp. 98–103. DOI: 10.1103/PhysRev.159.98. URL: https://link.aps.org/doi/10.1103/PhysRev.159.98.

[10] John Ziemer et al. 'Colloid microthruster flight performance results from space technology 7 disturbance reduction system'. In: (2017).

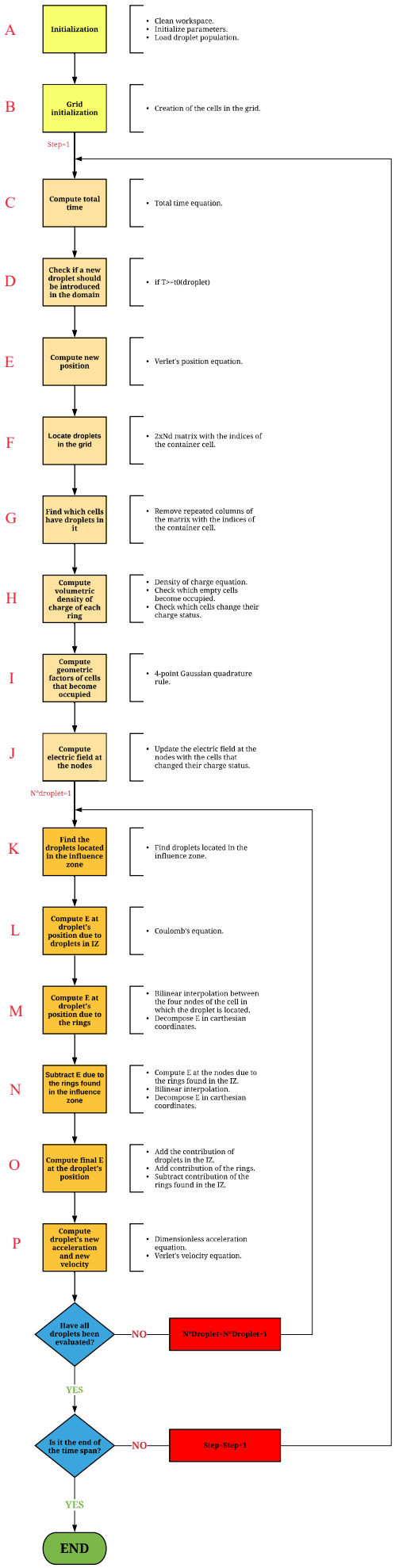
[11] Fredy R Zypman. 'Off-axis electric field of a ring of charge'. In: *American Journal of Physics* 74.4 (2006), pp. 295–300.

7 Annexes

7.1 Annex A: Flow diagram of the structure of the code

The flow diagram is in a non-standard page format.

Please find it attached right after this page.


A
Initialization
Clean workspace.
Initialize parameters.
Load droplet population.
B
Grid initialization
Creation of the cells in the grid.
Step=1
C
Compute total time
Total time equation.
D
Check if a new droplet should be introduced in the domain
if T>=t0(droplet)
E
Compute new position
Verlet's position equation.
F
Locate droplets in the grid
2xNd matrix with the indices of the container cell.
G
Find which cells have droplets in it
Remove repeated columns of the matrix with the indices of the container cell.
H
Compute volumetric density of charge of each ring
Density of charge equation.
Check which empty cells become occupied.
Check which cells change their charge status.
I
Compute geometric factors of cells that become occupied
4-point Gaussian quadrature rule.
J
Compute electric field at the nodes
Update the electric field at the nodes with the cells that changed their charge status.
N°droplet=1
K
Find the droplets located in the influence zone
Find droplets located in the influence zone.
L
Compute E at droplet's position due to droplets in IZ
Coulomb's equation.
M
Compute E at droplet's position due to the rings
Bilinear interpolation between the four nodes of the cell in which the droplet is located.
Decompose E in carthesian coordinates.
N
Subtract E due to the rings found in the influence zone
Compute E at the nodes due to the rings found in the IZ.
Bilinear interpolation.
Decompose E in carthesian coordinates.
O
Compute final E at the droplet's position
Add the contribution of droplets in the IZ.
Add contribution of the rings.
Subtract contribution of the rings found in the IZ.
P
Compute droplet's new acceleration and new velocity
Dimensionless acceleration equation.
Verlet's velocity equation.
Have all droplets been evaluated?
NO
N°Droplet=N°Droplet+1
YES
Is it the end of the time span?
NO
Step=Step+1
YES
END

7.2 Annex B: Environmental impact

This study has been carried out using only the computer, thereby, the environmental impact for it can be considered negligible or null.

7.3 Annex C: Budget

The costs of this study are mainly the labor costs and the license for the software. Such costs are detailed in the table below:

Item	**Cost**
Labors costs (300 h x 12$/h)	3600$
Matlab license	860$

Table 4: Budget

The total cost of the study is 4460$.

The costs associated with the experimental work have not been included due to the fact that the experimental work is not carried out with the intention of fulfilling the needs of this study. The experimental work is its own study and is much more complex than what has been mentioned in this thesis. The model only uses a part of the information that has been generated with the experimental set-up.

7.4 Annex D: Matlab code

Main code

```
%% MAIN SIMULATOR GENERAL MOTION OF DROPLETS
%% INITIALIZATION
clc
clear all
format long


tic

% Load Droplet population
load('DropletInitialCondition_Nd100.mat')


%Characterisitc parameters
%Specific charge (C/kg)
Qsc=549.24;
%Charge (C)
Qc=2.57329e-17;
%Lambda (m)
Lamc=4.1387e-8;
%Velocity (m)
Vc=643.3;

%Dimensionless coefficient for internal electric field
e0=8.854e-12; %Vaccum permitivity
PIi=Qsc*Qc/(4*pi*e0*Lamc*Vc^2);


N=length(Q); %Number of droplets in the population
step_Nd=zeros(1,N); %Step aat which the droplets is
```

```
    introduced
31
32 % Computation parameters initialization
33 delta_t=0.05;
34
35 T=0; %T is the absolute time during the simulation
36
37 % Number of steps in the simulation.
38 % The 1.2 factor is to allow some time after the last
    droplet comes out
39 N_steps=round(2*t0(length(t0))/delta_t);
40
41 %% Creation of a grid in cylindrical coordinates
42
43 global NA NR axial_nodes radial_nodes
44
45 %Number of axial divisions
46 NA=190;
47 %Thickness of each division
48 thickness_axial=zeros(1,NA);
49 thickness_axial(:)=30; %Set default value
50 %Number of radial divisions for each axial division
51 NR=zeros(1,NA);
52 NR(:)=70; %Set default value
53 %Thickness of radial divisions
54 thickness_radial=zeros(max(NR),NA);
55 thickness_radial(:,:)=8; %Set default value
56
57 %Generate axial and radial positions
58 %First division
59 %AXIAL
60 axial_nodes=zeros(1,NA);
```

```
61 %RADIAL
62 radial_nodes=zeros(max(NR),NA);
63 for j=2:1:NR(1)
64     radial_nodes(j,1)=radial_nodes(j-1,1)+thickness_radial(
       j,1);
65 end
66 %Rest of divisions
67 for i=2:1:NA
68     %Axial division
69     axial_nodes(i)=axial_nodes(i-1)+thickness_axial(i);
70     %Radial division
71     for j=2:1:NR(i)
72         radial_nodes(j,i)=radial_nodes(j-1,i)+
           thickness_radial(j,i);
73     end
74 end
75
76 % Initialize density of charge matrix
77 rhoQ=zeros(max(NR),NA);
78
79 %% VERLET
80
81 %Initialize motion variables
82 X=zeros(N,N_steps);
83 Y=zeros(N,N_steps);
84 Z=zeros(N,N_steps);
85 Vx=zeros(N,N_steps);
86 Vy=zeros(N,N_steps);
87 Vz=zeros(N,N_steps);
88
89 %Acceleration due to internal electric field
90 ax_i=zeros(N,N_steps);
```

```
91  ay_i=zeros(N,N_steps);
92  az_i=zeros(N,N_steps);
93
94  %Acceleration due to external electric field
95  ax_e=zeros(N,N_steps);
96  ay_e=zeros(N,N_steps);
97  az_e=zeros(N,N_steps);
98
99  %Total acceleration
100 ax=zeros(N,N_steps);
101 ay=zeros(N,N_steps);
102 az=zeros(N,N_steps);
103
104 %Load first droplet
105 Nd=1; %Nd accounts for the number of droplets that are out
106 step_Nd(1)=1;
107
108 X(1,1)=x0(1);
109 Y(1,1)=y0(1);
110 Z(1,1)=z0(1);
111
112 Vz(1,1)=vz0(1);
113
114
115 for step=1:1:N_steps-1
116     T=step*delta_t; %Absolute time
117
118
119     if Nd<N
120         if t0(Nd+1)<=T
121             %When the absolute time reaches the initiation
                time of the new
```

```
122             %droplet, a new droplet comes out
123             Nd=Nd+1;
124             fprintf('Number of droplets out: %.0f \n',Nd)
125             X(Nd,step)=x0(Nd);
126             Y(Nd,step)=y0(Nd);
127             Z(Nd,step)=z0(Nd);
128
129             Vz(Nd,step)=vz0(Nd);
130
131             step_Nd(Nd)=step;
132         end
133     end
134
135     %_____________________________________
136     % STARTING VERLET
137     %_____________________________________
138
139     for i=1:1:Nd
140
141         %Computation of the new position
142         X(i,step+1)=X(i,step)+Vx(i,step)*delta_t+0.5*ax(i,
                step)*delta_t^2;
143         Y(i,step+1)=Y(i,step)+Vy(i,step)*delta_t+0.5*ay(i,
                step)*delta_t^2;
144         Z(i,step+1)=Z(i,step)+Vz(i,step)*delta_t+0.5*az(i,
                step)*delta_t^2;
145
146         %Computation of intermediate velocity
147         int_Vx(i)=Vx(i,step)+0.5*ax(i,step)*delta_t;
148         int_Vy(i)=Vy(i,step)+0.5*ay(i,step)*delta_t;
149         int_Vz(i)=Vz(i,step)+0.5*az(i,step)*delta_t;
150     end
```

```
151
152     %Locate Droplets in the grid
153     droplet_position= FIND_droplets_position (axial_nodes,
            radial_nodes,NA,NR,Nd,step+1,X,Y,Z);
154
155     %Find cells with droplets
156     cells_with_D=unique(droplet_position,'rows');
157
158     %Compute the density of charge of the cells with
            droplets and the cells
159     %that have gone from empty to occupied
160     [rhoQ,cells_changed,change]=density_of_charge(
            cells_with_D,Nd,rhoQ,Q,droplet_position);
161
162     %Geometric factor computation at all nodes from cells
            that are occupied
163     if change
164         [geometric_factor]=geometric_value(cells_changed);
165     end
166     %Compute axial and radial E at the nodes
167     if exist('geometric_factor')==1
168         [E_nodes]=Compute_E_nodes(geometric_factor,rhoQ,
                cells_with_D);
169     end
170     %Zoneof influence will be defined by IZ axial divisions
            right and left
171     %of the droplet
172     IZ=3;
173
174     Ex_r=0;
175     Ey_r=0;
176     Ez_r=0;
```

```
177     for i=1:1:Nd
178
179         %Indeces of the cell in which the droplet is
                located
180         iA=droplet_position(i,1);
181         iR=droplet_position(i,2);
182         %Find droplets in the IZ
183         D_IZ=droplets_in_IZ(Nd,i,IZ,droplet_position);
184
185         %Computation of internal electric field (DROPLETS)
186         [Ex_i,Ey_i,Ez_i]=internal_electric_field_ONE_D_GRID
                (i,D_IZ,step+1,X,Y,Z,Q);
187
188         %Computation of internal electric field (RING)
189         if exist('E_nodes')==1
190             [Ex_r,Ey_r,Ez_r]=bilinear_interpolation(iA,iR,
                    E_nodes,X(i,step),Y(i,step),Z(i,step));
191         end
192
193         %Computation of the electric field of the rings in
                the IZ
194         for k=1:1:size(cells_with_D,1)
195             if i>(iA-IZ) || i<=(iA+IZ)
196                 [Ex_g,Ey_g,Ez_g]=E_ring_gauss(Nd,i,step+1,
                        cells_with_D(k,1),cells_with_D(k,2),X,Y,
                        Z,Q,axial_nodes,radial_nodes,
                        droplet_position);
197                 Ex_i=Ex_i-Ex_g;
198                 Ey_i=Ey_i-Ey_g;
199                 Ez_i=Ez_i-Ez_g;
200             end
201         end
```

```
202
203             %Computation of each dropplet's acc due to internal
                    E field
204             ax_i(i,step+1)=PIi*Qs(i)*(Ex_i+Ex_r);
205             ay_i(i,step+1)=PIi*Qs(i)*(Ey_i+Ey_r);
206             az_i(i,step+1)=PIi*Qs(i)*(Ez_i+Ez_r);
207
208             %Computation of total acceleration
209             ax(i,step+1)=ax_i(i,step+1)+ax_e(i,step+1);
210             ay(i,step+1)=ay_i(i,step+1)+ay_e(i,step+1);
211             az(i,step+1)=az_i(i,step+1)+az_e(i,step+1);
212
213             %Computation of velocity
214             Vx(i,step+1)=int_Vx(i)+0.5*ax(i,step+1)*delta_t;
215             Vy(i,step+1)=int_Vy(i)+0.5*ay(i,step+1)*delta_t;
216             Vz(i,step+1)=int_Vz(i)+0.5*az(i,step+1)*delta_t;
217         end
218
219         %_____________________________________
220         %ENDING VERLET
221         %_____________________________________
222
223         %_____________________________________
224         %ENERGY BALANCE
225         %_____________________________________
226         %TO BE IMPLEMENTED!!!!!!!!!
227
228
229
230
231 end
232 toc
```

```
233
234 %% Plot results
235
236
237 %SAVE set as true will enable saving the data in the
        current folder
238 SAVE1=true;
239
240
241 %Compute radial component
242 R=sqrt(X(:,:).^2+Y(:,:).^2);
243 Rmax=max(max(R));
244 Rmin=min(min(R));
245
246 if SAVE1
247     save DATA
248     %Time frames
249     TF=round(N_steps*(0.01:0.1:1));
250     for i=1:1:length(TF)
251         %plot3_motion(TF(i),delta_t,X,Y,Z)
252         %plot2_motion(TF(i),delta_t,X,Y,Z)
253         plotr_motion(SAVE1,TF(i),delta_t,Nd,R,Z,Rmax,Rmin)
254     end
255     close all
256 end
257
258 %% Energy balance
259 %Energy computation
260 SAVE2=true;
261 for i=1:1:N_steps
262
263     EP(i)=potential_energy(N,X*Lamc,Y*Lamc,Z*Lamc,Q*Qc,i);
```

```
264     EC(i)=kinetic_energy (Vx*Vc,Vy*Vc,Vz*Vc,Qs*Qsc,Q*Qc,i);
265     E(i)=EP(i)+EC(i);
266 end
267
268 %Plot results
269 t=delta_t*(1:1:N_steps);
270
271 fig2=figure('Name','Energy balance');
272 plot(t,EC,'.',t,EP,'.',t,E,'.',delta_t*step_Nd,E(step_Nd'),
      'k*')
273 xlabel('Time')
274 ylabel('Energy')
275 title('Energy balance')
276 legend('Kinetic','Potential','Mechanical')
277
278 if SAVE2
279     saveas(fig2,'Energy_balance')
280     saveas(fig2,'Energy_balance','png')
281 end
```

Functions

```
1 function droplet_position= FIND_droplets_position (
     axial_division,radial_division,NA,NR,Nd,step,X,Y,Z)
2 droplet_position=zeros(Nd,2);
3 %Compute radial position
4 R=sqrt(X(:,step).^2+Y(:,step).^2);
5
6 for i=1:1:Nd
7     axial_found=false;
8     radial_found=false;
9     for j=2:1:NA
10        if Z(i,step)<axial_division(j) && axial_found==
             false
```

```
11                droplet_position(i,1)=j-1; %Axial position
12                axial_found=true;
13                for k=2:1:NR
14                    if R(i)<radial_division(k) && radial_found
                          ==false
15                        droplet_position(i,2)=k-1;
16                        radial_found=true;
17                        break
18                    end
19                end
20                if radial_found
21                    break
22                end
23            end
24        end
25        if radial_found == false
26            fprintf('Radial position=%.4f    Axial position=%.4f
                  \n',R(i),Z(i,step));
27            error('Droplet %.0f has not been located in the
                  grid ',i)
28        end
29    end
30
31    end
32
33    function [rhoQ,cells_changed,change]=density_of_charge(
          cells_with_D,Nd,rhoQ,Q,droplet_position)
34    global axial_nodes radial_nodes
35    counter=1; %Counter for the cells that change.
36    for i=1:1:size(cells_with_D,1)
37
38        %Indices of the cell
```

```
39      iA=cells_with_D(i,1);
40      iR=cells_with_D(i,2);
41
42      %Droplets in the cell
43      droplets = droplets_ring (Nd,droplet_position,iA,iR);
44
45      %Limits of the integration
46      if iA==1
47          aA=0;
48      else
49          aA=axial_nodes(iA-1);
50      end
51      bA=axial_nodes(iA);
52
53      if iR==1
54          aR=0;
55      else
56          aR=radial_nodes(iR-1);
57      end
58      bR=radial_nodes(iR);
59
60      %Volume of the cell
61      Vol=pi*(bA-aA)*(bR^2-aR^2);
62
63      Q_ring=0;
64      for k=1:1:length(droplets)
65          Q_ring=Q_ring+Q(droplets(k));
66      end
67      rhoQ_raw=Q_ring/Vol;
68
69      rhoQ(iA,iR)=rhoQ_raw;
70
```

```
71      %Check if the cell has gone frome empty to occupied
72      if rhoQ_raw~=rhoQ(iA,iR) && rhoQ(iA,iR)==0
73          %the cell has gone from empty to occupied
74          cells_changed(counter,1)=iA;
75          cells_changed(counter,2)=iR;
76          counter=counter+1;
77          change=true;
78          %add the cell to the list with cells that have
                changed their charge
79      end
80  end
81  if counter==1
82      cells_changed=[];
83      change=false;
84  end
85  end
86
87
88  function droplets = droplets_ring (Nd,droplet_position,
        Axial_index,Radial_index)
89  N=1; %Number of droplets in the ring -1
90  for i=1:1:Nd
91      if droplet_position(i,1)==Axial_index &&
           droplet_position(i,2)==Radial_index
92          droplets(N)=i;
93          N=N+1;
94      end
95  end
96
97  end
98
99  function [geometric_factor]=geometric_value(cells_changed)
```

```
100
101 global axial_nodes radial_nodes
102 for I=1:1:size(cells_changed,1)
103
104     %indices of the cell
105     iA=cells_changed(I,1);
106     iR=cells_changed(I,2);
107
108     %Gauss Quadrature 4 points value
109     Wg=[0.3478548451374539, 0.6521451548625463,
            0.6521451548625463, 0.3478548451374539];
110     Xg=[-0.8611363115940526, -0.3399810435848563,
            0.3399810435848563, 0.8611363115940526];
111
112     %Limits of the integration
113     if iA==1
114         aA=0;
115     else
116         aA=axial_nodes(iA-1);
117     end
118     bA=axial_nodes(iA);
119
120     if iR==1
121         aR=0;
122     else
123         aR=radial_nodes(iR-1);
124     end
125     bR=radial_nodes(iR);
126
127     for J=1:1:length(axial_nodes)
128         for K=1:1:length(radial_nodes(:,J))
129             %Don't compute the geometric factor at the
```

```
                    nodes of the cell
130             %itself
131             if J~=iA && K~=iR
132                 %Compute E using Gauss quadrature
133                 Er=0;
134                 Ez=0;
135
136                 for i=1:1:4 %Axial
137                     for j=1:1:4 %Radial
138                         Ri=(bR-aR)*(Xg(j))/2+(bR+aR)/2;
139                         Zi=(bA-aA)*(Xg(i))/2+(bA+aA)/2;
140                         %Density of charge is set to 1
                                because it will be multiplied
                                later
141                         [Eri, Ezi]=E_phi_ring (1,Ri,Zi,0,
                                radial_nodes(K,J),axial_nodes(J)
                                );
142                         Er=Er+Eri*Wg(j)*Wg(i)*(bR-aR)/2*(bA
                                -aA)/2;
143                         Ez=Ez+Ezi*Wg(j)*Wg(i)*(bR-aR)/2*(bA
                                -aA)/2;
144                     end
145                 end
146
147                 geometric_factor(J,K).axial(iA,iR)=Ez;
148                 geometric_factor(J,K).radial(iA,iR)=Er;
149             end
150         end
151     end
152 end
153
154 end
```

```
155
156 function [E_nodes]=Compute_E_nodes(geometric_value,rhoQ,
      cells_with_D)
157 global NA NR
158
159 %For all nodes with droplets in it:
160 %Cells_with_D have the indices of only two nodes. We need
      to have the
161 %Electric field in the four nodes of the cell.
162 Nc=size(cells_with_D,1);
163 for i=1:1:Nc
164     iA=cells_with_D(i,1);
165     iR=cells_with_D(i,2);
166
167     %Add the 1st node missing
168     cells_with_D(Nc+1+3*(i-1),1)=iA;
169     cells_with_D(Nc+1+3*(i-1),2)=iR-1;
170     %Add the 1st node missing
171     cells_with_D(Nc+2+3*(i-1),1)=iA-1;
172     cells_with_D(Nc+2+3*(i-1),2)=iR-1;
173     %Add the 1st node missing
174     cells_with_D(Nc+3+3*(i-1),1)=iA-1;
175     cells_with_D(Nc+3+3*(i-1),2)=iR;
176 end
177
178 cells_with_D=unique(cells_with_D,'rows');
179
180 for i=1:1:size(cells_with_D,1)
181
182     iA=cells_with_D(i,1);
183     iR=cells_with_D(i,2);
184
```

```
185         axial_E=0;
186         radial_E=0;
187         %Add the contribution of each cell
188         for I=1:1:NA
189             for J=1:1:NR(I)
190                 if rhoQ(I,J)~=0
191                     axial_E=axial_E+geometric_value(iA,iR).
                            axial(I,J)*rhoQ(I,J);
192                     radial_E=radial_E+geometric_value(iA,iR).
                            radial(I,J)*rhoQ(I,J);
193                 end
194             end
195         end
196         E_nodes(iA,iR).axial=axial_E;
197         E_nodes(iA,iR).radial=radial_E;
198
199 end
200 end
201
202 function D_IZ=droplets_in_IZ(Nd,iD,IZ,droplet_position)
203
204 %index of the cell in which the droplet is located
205 iA=droplet_position(iD,1);
206 D_IZ=[];
207 for i=iA:-1:(iA-IZ+1)
208     if i>=1
209         droplets_in_i=droplets_axial_division (Nd,
                droplet_position,i);
210         D_IZ=[D_IZ droplets_in_i];
211     end
212 end
213
```

```
214 for i=iA:+1:(iA+IZ)
215     droplets_in_i=droplets_axial_division (Nd,
            droplet_position,i);
216     D_IZ=[D_IZ droplets_in_i];
217 end
218 end
219
220 function droplets = droplets_axial_division (Nd,
        droplet_position,Axial_index)
221 N=1; %Number of droplets in the ring -1
222 for i=1:1:Nd
223     if droplet_position(i,1)==Axial_index
224         droplets(N)=i;
225         N=N+1;
226     end
227 end
228 if N==1
229     droplets=[];
230 end
231 end
232
233 function [Ex,Ey,Ez]=internal_electric_field_ONE_D_GRID (iD,
        D_IZ,Nstep,X,Y,Z,Q)
234 %{
235 The function gives the internal electric field that each
        particle feels.
236 %}
237 Ex=0;
238 Ey=0;
239 Ez=0;
240 for k=1:1:length(D_IZ)
241     j=D_IZ(k);
```

```
242         if  j~=iD
243             %Compute distance between particles
244             r=sqrt((X(iD,Nstep)-X(j,Nstep))^2+(Y(iD,Nstep)-Y(j,
                    Nstep))^2+(Z(iD,Nstep)-Z(j,Nstep))^2);
245
246             Ex=Ex+Q(j)*(X(iD,Nstep)-X(j,Nstep))/r^3;
247             Ey=Ey+Q(j)*(Y(iD,Nstep)-Y(j,Nstep))/r^3;
248             Ez=Ez+Q(j)*(Z(iD,Nstep)-Z(j,Nstep))/r^3;
249         end
250     end
251
252     end
253
254     function [Ex_r,Ey_r,Ez_r]=bilinear_interpolation(iA,iR,
            E_nodes,x0,y0,z)
255
256     %Radial position of the droplet
257     r=sqrt(x0^2+y0^2);
258
259     global radial_nodes axial_nodes
260     r1=radial_nodes(iR);
261     r2=radial_nodes(iR+1);
262     z1=axial_nodes(iA);
263     z2=axial_nodes(iA+1);
264
265     %AXIAL
266     f11=E_nodes(iA-1,iR-1).axial;
267     f12=E_nodes(iA-1,iR).axial;
268     f21=E_nodes(iA,iR-1).axial;
269     f22=E_nodes(iA,iR).axial;
270
271     A=1/((z2-z1)*(r2-r1));
```

```
272 A1=f11*(z2-z);(r2-r);
273 A2=f21*(z-z1);(r2-r);
274 A3=f12*(z2-z);(r-r1);
275 A4=f22*(z-z1);(r-r1);
276
277 Ez_r=A*(A1+A2+A3+A4);
278
279 %RADIAL
280 f11=E_nodes(iA-1,iR-1).radial;
281 f12=E_nodes(iA-1,iR).radial;
282 f21=E_nodes(iA,iR-1).radial;
283 f22=E_nodes(iA,iR).radial;
284
285 A=1/((z2-z1)*(r2-r1));
286 A1=f11*(z2-z);(r2-r);
287 A2=f21*(z-z1);(r2-r);
288 A3=f12*(z2-z);(r-r1);
289 A4=f22*(z-z1);(r-r1);
290
291 Er=A*(A1+A2+A3+A4);
292
293 %Convert from cylindrical to cartesian
294 theta=atan(y0/x0);
295 if X(iD,step)>0
296     Ex_r=-Er*cos(theta);
297     Ey_r=-Er*sin(theta);
298 else
299     Ex_r=Er*cos(theta);
300     Ey_r=Er*sin(theta);
301 end
302
303
```

```
304 end
305
306 function [Ex,Ey,Ez]=E_ring_gauss(Nd,iD,step,iA,iR,X,Y,Z,Q,
        axial_division,radial_division,droplet_position)
307
308 %Gauss Quadrature 4 points value
309 Wg=[0.3478548451374539, 0.6521451548625463,
        0.6521451548625463, 0.3478548451374539];
310 Xg=[-0.8611363115940526, -0.3399810435848563,
        0.3399810435848563, 0.8611363115940526];
311
312 %Limits of the integration
313 if iA==1
314     aA=0;
315 else
316 aA=axial_division(iA-1);
317 end
318 bA=axial_division(iA);
319
320 if iR==1
321     aR=0;
322 else
323     aR=radial_division(iR-1);
324 end
325 bR=radial_division(iR);
326
327 %Find volumetric density of charge
328
329 %Find Volume of the ring
330 Vol=pi*(bA-aA)*(bR^2-aR^2);
331
332 droplets = droplets_ring (Nd,droplet_position,iA,iR);
```

```
333 Q_ring=0;
334 for k=1:1:length(droplets)
335     Q_ring=Q_ring+Q(droplets(k));
336 end
337 rhoQ=Q_ring/Vol;
338 %Compute E using Gauss quadrature
339 Er=0;
340 Ez=0;
341 for i=1:1:4 %Axial
342     for j=1:1:4 %Radial
343         Ri=(bR-aR)*(Xg(j))/2+(bR+aR)/2;
344         Zi=(bA-aA)*(Xg(i))/2+(bA+aA)/2;
345
346         [Eri, Ezi]=E_phi_ring (rhoQ,Ri,Zi,X(iD,step),Y(iD,
                step),Z(iD,step));
347         Er=Er+Eri*Wg(j)*Wg(i)*(bR-aR)/2*(bA-aA)/2;
348         Ez=Ez+Ezi*Wg(j)*Wg(i)*(bR-aR)/2*(bA-aA)/2;
349     end
350 end
351 %Convert from cylindrical to cartesian
352 theta=atan(Y(iD,step)/X(iD,step));
353 if X(iD,step)>0
354     Ex=-Er*cos(theta);
355     Ey=-Er*sin(theta);
356 else
357     Ex=Er*cos(theta);
358     Ey=Er*sin(theta);
359 end
360 end
361
362 function plotr_motion (SAVE,step,delta_t,Nd,R,Z,Rmax,Rmin)
363     Title=sprintf('T=%.4f',(step*delta_t));
```

```
364         Subtitle=sprintf('Delta_t=%.4f       N droplets=%.0f',
                delta_t,Nd);
365         fig=figure ('Name',Title);
366         plot(Z(:,step),R(:,step),'.')
367         xlabel('Z')
368         ylabel('R')
369         xlim([0 max(Z(:,length(Z)))])
370         ylim([Rmin Rmax])
371         title({Title,Subtitle})
372
373         if SAVE
374             saveas(fig,sprintf('%.0f.png',(step*delta_t)*1000),
                    'png')
375         end
376
377 end
378
379 function [PE] = potential_energy(Nd,X,Y,Z,Q,Nstep)
380 e0=8.854e-12;
381 PE=0;
382
383 %Number of droplets in the domain at the specified time
        step
384 Nd=length(nonzeros(X(:,Nstep)));
385 for i=1:1:Nd
386     P=0; %electric potential due to all point charges
387     for j=1:1:Nd
388         if j~=i
389             %Compute distance between particles
390             r=sqrt((X(i,Nstep)-X(j,Nstep))^2+(Y(i,Nstep)-Y(
                    j,Nstep))^2+(Z(i,Nstep)-Z(j,Nstep))^2);
391             P=P+Q(j)/(r);
```

```
392             end
393         end
394         PE=PE+Q(i)*P;
395     end
396     PE=PE/(2*4*e0*pi);
397     %PE=PE/(2);
398     end
399
400     function EC=kinetic_energy (vx,vy,vz,Qs,Q,Nstep)
401     V=[vx(:,Nstep) vy(:,Nstep) vz(:,Nstep)];
402     M=Q./Qs;
403     %Compute kintetic energy of each droplet
404     Ec=0.5*(sqrt(V(:,1).^2+V(:,2).^2+V(:,3).^2)).^2;
405     Ec=M.*Ec;
406     %Compute kinetic energy of the whole system
407     EC=sum(Ec);
408     end
```